JPL Publication D-65838

DARPA DTN Phase 3 Core Engineering Support

Final Report

Prepared by

J. L. Torgerson, Principal Investigator

Prepared for

Defense Advanced Research Projects Agency
by

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

30 April 2010

Acknowledgements

This report was prepared by the DARPA DTN team at JPL, which included, Richard Borgen, James McKelvey, John Segui, Leigh Torgerson, Phil Tsao, and excellent hardware support from the Joshua Schoolcraft of the JPL Protocol Technology Lab.

We would like to thank Preston Marshall for his long-time support of DTN, Jim Snyder for his guidance and enthusiasm for the USMC tests, and Vint Cerf for his inspiration and leadership over the years as the DTN protocols were developed.

This task put JPL in the middle of a very complex integration effort that involved excellent teams from BBN, MITRE and the people at SPAWAR San Diego, and it was a true pleasure working with the dedicated professionals from these organizations.

Finally, we thank the United States Marine Corps and the fine people we worked with at MCBH and MCTSSA. Their support and enthusiasm for this DTN project was invaluable, and we thank them for their service to the country.

TABLE OF CONTENTS

1 Executive Summary

This report covers the initial DARPA DTN Phase 3 activities as JPL provided Core Engineering Support to the DARPA DTN Program, and then further details the culmination of the Phase 3 Program with a systematic development, integration and test of a disruption-tolerant C2 Situation Awareness (SA) system that may be transitioned to the USMC and deployed in the near future. The system developed and tested was a SPAWAR/JPL-developed Common Operating Picture Fusion Tool called the Software Interoperability Environment (SIE), running over Disruption Tolerant Networking (DTN) protocols provided by BBN and MITRE, which effectively extends the operational range of SIE from normal fully-connected internet environments to the mobile tactical edges of the battlefield network.

The field and laboratory tests conducted showed that DTN provides significant advantages by enabling 100% of the SA data to be delivered to mobile, often disconnected laptops, as contrasted with the traditional internet-based systems that failed to provide the necessary service to the tactical edge network.

JPL acted as the integrators of the system, and supported SPAWAR in their conduct of the tests. SPAWAR, BBN and MITRE did the detailed analysis of their portions of the protocol stack. This report summarizes the tests, discusses qualitative results, our observations, and provides details of the JPL lab configuration, Qualnet simulation software models and SharedNet log analysis tools.

2 Background

This report details the scope, activities and technical results achieved by the Jet Propulsion Laboratory under Task Order 98-9538. This Task Order enabled JPL to provide DARPA with Core Engineering Support for Disruption Tolerant Networking activities in Phase 3 of the DARPA DTN program.

Phase 3 of this program addressed the integration of the Phase 2 technologies into a single framework that meets military needs, and prepared for large-scale demonstrations of the DTN technology.

During Phase 2, the DTN specifications matured, a stable reference implementation of the DTN bundle protocol was developed, and routing and security mechanisms were tested in laboratory conditions and small-scale field demonstrations.

During Phase 3, JPL continued work on protocol standardization activities with the DTN Research Group (DTNRG), investigated various simulation methods to enable

large-scale assessment of routing algorithms, and integrated and tested a DTN-based Command and Control (C2) system for use by the USMC in tactical environments. This integration effort combined software written during Phase 3 by BBN and MITRE, JPL-developed SharedNet software and a Software Interoperability Environment system developed by the Space and Naval Warfare Systems Center, San Diego (SSC San Diego, or SPAWAR.)

This final report will cover both the standardization and simulation activities in the early part of Phase 3, as well as detail the integration and test activities in support of the USMC C2 system.

The testing and demonstration of the USMC C2 system was undertaken with the full participation of SPAWAR, BBN and MITRE. The overall results of the system field and laboratory testing and comparison to legacy systems will be written by SPAWAR. The performance of the DTN Bundle Protocol Agent, its DR routing algorithms and the effectiveness of the NACK-Oriented Reliable Multicast (NORM) layer will be reported on in detail by the SPAWAR, BBN and MITRE teams.

This report will focus on the JPL integration and test activities (in particular the Protocol Technology Lab configuration and support for field and lab testing), and the overall observed benefits of DTN in the tests we conducted.

2.1 Description of Work

The work conducted for DAPRA during Phase 3 of the DARPA DTN program was based on the following items in the Task Order:

3.1.1 TO 98-9538 Scope of Work – Early Part of Phase 3 (Commencing September 2008)

The original Phase 3 tasks were to:

a) Continue to support the development of the controlling DTN architectural and protocol specifications and thus guide the overall progressive evolution of the technology.
b) Participate in the continued testing and development of the DTN API specification being conducted under MITRE leadership.
c) Maintain Header Compression software, and develop an RFC for DTN HC.
d) Maintain DARPA-DTN version of the JPL SDR and AMS software and provide assistance in its use.

e) Provide a large-scale simulation of hundreds of DTN nodes on a supercomputer cluster for validation/ scalability checks of routing methods planned for Phase 3 demos.

3.1.2 TO 98-9538 Scope of Work – Final Part of Phase 3 (Commencing June 2009)

In Summer of 2009, the Task Order was revised to focus on the SPAWAR demonstration & tests. The salient tasks included:

a) The Core Engineering Support team will support the Space and Naval Warfare Systems (SPAWAR) Command demonstration of the Software Integration Environment / Shared Net C2 system (SIE/SN) over the DTN network.
b) Integration of the JPL SharedNet software with BBN's BPA DTN software during Spiral 1 (software integration)
c) Test operation of integrated SIE/SN/BPA software in multimode topologies to emulate the tactical environment to be demonstrated by SPAWAR during Spiral 2 (Field testing).
d) Assist SPAWAR in the integration and testing in their lab and in the field environment during Spiral 2.
e) Assist SPAWAR in Marine Corps Tactical Systems Support Activity (MCTSSA) testing and final operational reports to document the use of DTN and the performance of the SIE/SN/BPA software in the tactical environment.

3 Results and Accomplishments

3.1 Early Phase 3 work

3.1.1 Accomplishments - Early Part of Phase 3 (Commencing September 2008)

a) During Phase 3, JPL authored/coauthored the following Internet Research Task Force RFCs as part of the DTN Standardization effort:

RFC 5325 - Licklider Transmission Protocol – Motivation

RFC 5326 - Licklider Transmission Protocol – Specification

RFC 5327 - Licklider Transmission Protocol – Security Extensions

In addition, Internet Drafts were published to include:

irtf-dtnrg-cbhe-04 – Compressed Bundle Header Encoding

draft-burleigh-dtnrg-cgr-00 - Contact Graph Routing

draft-irtf-dtnrg-ecos-00 - Bundle Protocol Extended Class Of Service (ECOS)

draft-irtf-dtnrg-dtn-uri-scheme-00 - The DTN URI Scheme

b) During Phase 3, JPL participated in technical test and development of the DTN APIs as implemented by the DTNRG DTN2, BBN's BPA, and the JPL Interplanetary Overlay (ION) DTN bundle agent, and interoperability testing was conducted. This included participation in the July 29-30, 2009 DTNRG DisConnectathon.

c) JPL developed and maintained the Header Compression software known as CBHE. Compressed Bundle Header Encoding (CBHE) is a convention by which Delay-Tolerant Networking (DTN) Bundle Protocol (BP) [RFC5050] "convergence-layer adapters" may represent endpoint identifiers in a compressed form within the primary blocks of bundles, provided those endpoint identifiers conform to the structure prescribed by this convention. It enables the "dictionary" of ASCII-text endpoint ID strings to be omitted altogether from the primary block of a bundle, greatly reducing the size of the primary block. This reduction in bundle size can be significant, especially when the bundle's payload is relatively small, as is anticipated for a number of DTN applications such as space flight operations (and as is in any case true of bundles carrying BP administrative records). The initial Internet Draft for CBHE was posted in 2006. Since then the specification has been refined several times; the final Internet Draft for CBHE is currently in "IRSG poll" state, pending approval for its submission as an experimental RFC.

d) JPL provided Simple Data Recorder (SDR) and Asynchronous Messaging System (AMS) implementations to the DTNRG with APIs for DTN2. However, practically speaking, the community interest in SDR and AMS has been limited to spacecraft applications, and the flight-qualified version of DTN (ION) is currently the implementation of choice for users of SDR or AMS.

e) Work was commenced on the use of the JPL supercomputer cluster for large-scale node deployment for the purpose of simulation and routing algorithm scalability. Initial work with the supercomputer cluster showed that the practicality of using this particular asset was less than desired. The nodes and their IP addresses (which are needed for the routing tables and various convergence layers in most of the common DTN implementations) were dynamically assigned by the SC system at run-time, and this was difficult to cope with. In addition, the control and simulation of link outages and topology changes would have required a level of access to routing tables in the SC cluster that was not allowable. Simulation of links with additional Qualnet

simulations was investigated, and the overall conclusion was that it was much more practical and cost-effective to abandon the SC cluster work in favor of large scale Qualnet simulations.

Simulating large-scale operations and different routing algorithms in Qualnet has the added advantage that rather than requiring actual routing algorithm code (as would be needed in the SC cluster), Qualnet allows for algorithmic simulation when actual code is not yet developed.

Shortly after the initial implementation and feasibility assessments of this work item, the Task Plan was changed to focus on the SPAWAR/USMC demonstration and test work.

3.2 Final Phase 3 Work

3.2.1 Changes from Final Version of the Task Plan

As described in section 1.1.2, the final part of Phase 3 was to focus on assisting SPAWAR in demonstrating its SIE/SN software over DTN. This included providing a local integration environment wherein a SPAWAR-funded JPL programmer could integrate the SharedNet software with the BBN-provided Bundle Protocol Agent (BPA) software on Linux-based platforms. There followed several large change of scope which greatly increased the complexity and technical difficulty of the task:

It was discovered that the USMC laptops that would be used in the lab and field demonstrations were required to run Windows XP rather than native Linux. Even though the tests were to be a demonstration of the efficacy of DTN in the tactical environment, the operational USMC networks we were planning to use would not permit native Linux operation, and the SECNET11 wireless cards planned for use only had Windows drivers in any case. Since BBN did not have a Windows implementation of BPA, the integration had to be done using a VMWare Virtual Machine Linux environment, with SIE/SharedNet running in the Windows environment, with a BBN-supplied Java to Linux VM program to allow the SIE/SN software to communicate with the Linux BPA.

This had technical ramifications that included the need to accommodate multiple IP addresses (one set for Windows, one set for Linux), and some complex work to get the time synchronization to work properly between Windows and the Linux VM.

The difficulties faced with getting the time to synchronize properly between the Windows and Linux VM operating systems led us to use a non-standard BPA that was developed by BBN that used relative bundle expiration time instead of absolute

bundle expiration time. The BPA implementation provided worked well in this regard, but it is not interoperable with any other DTNRG or NASA DTN implementation. This issue is currently being addressed in the IRTF, and a plan on how to standardize relative time DTN implementations was developed during the March 25th, 2010 DTNRG meeting at the Anaheim IETF meeting.

In addition, the DARPA BBN / MITRE team decided the best approach for the tests would be to use the Naval Research Lab's NACK-Oriented Reliable Multicast (NORM), using a convergence layer adapter (CLA) written by MITRE Corp.

While this promised to save bandwidth on the wireless links, the integration between BBN's IP Neighbor Discovery / Disrupted Routing mechanisms and the MITRE CLA/NORM software proved difficult as well. Inasmuch as both the BBN software and NORM have reliability mechanisms built in, the tuning of those mechanisms was time-consuming, and required several on-site visits of both BBN and MITRE engineers in the Protocol Technology Lab at JPL to work out the interfaces.

Shortly before the December 2009 Dry Run of the field tests at the USMC Base in Kaneohe Bay, Hawaii, some additional complexity to the tests was added to the tests:

a) Addition of Northrop-Grumman's C2PC as the user application on top of SIE/SN rather than using the SIE/SN Store Browser application

b) Addition of NRL's Simple Multicast Forwarding (SMF) to wireless nodes in field testing to enable a quantitative comparison of a MANET-like forwarder without store-and-forward against DTN

After the Field Testing at USMC Base Hawaii, the original plan was for SPAWAR to set up additional lab testing at the Marine Corps Tactical Support Systems Activity (MCTSSA) labs at Camp Pendleton, for detailed evaluation by the MCTSSA staff and the MCTSSA Chief Engineer for Tactical Networks. MCTSSA had a scheduling conflict with the planned tests, and the two weeks of lab tests were moved to the JPL PTL (aided by some additional funding provided by SPAWAR.)

During the week before the arrival of the DARPA-directed addition of tests for parallel testing of native C2PC to compare against SIE/DTN, we were directed by DARPA to add a second satellite hop to the topology, to use DARPA-furnished Apposite Linktrophy 4500 satellite channel emulators, and to provide for a separate test string to enable MCTSSA to evaluate a Performance Enhancing Proxy known as WARP with DTN.

All of these changes to the original task plan were accomplished successfully with no impact to the test schedule, but the net result was that while all of this software was finally successfully integrated and DTN successfully demonstrated, the additional scope and time it took to accomplish left somewhat inadequate time for thorough end-to-end testing and exploration of data collection and analysis techniques before the field and lab tests.

3.2.2 Integration Tasks Accomplished

3.2.2.1 Overall System Software Protocol Stack

SIE with SharedNet are Java-based applications that can run in either Windows or Linux. The USMC field laptops run Windows XP only, but the BBN and MITRE DTN/NORM protocols only run in Linux. So a hybrid system of Windows and Linux machines had to be developed, with some machines running Linux only, with others running Windows XP and hosting VMWare virtual machines (VMs) running Ubuntu Linux.

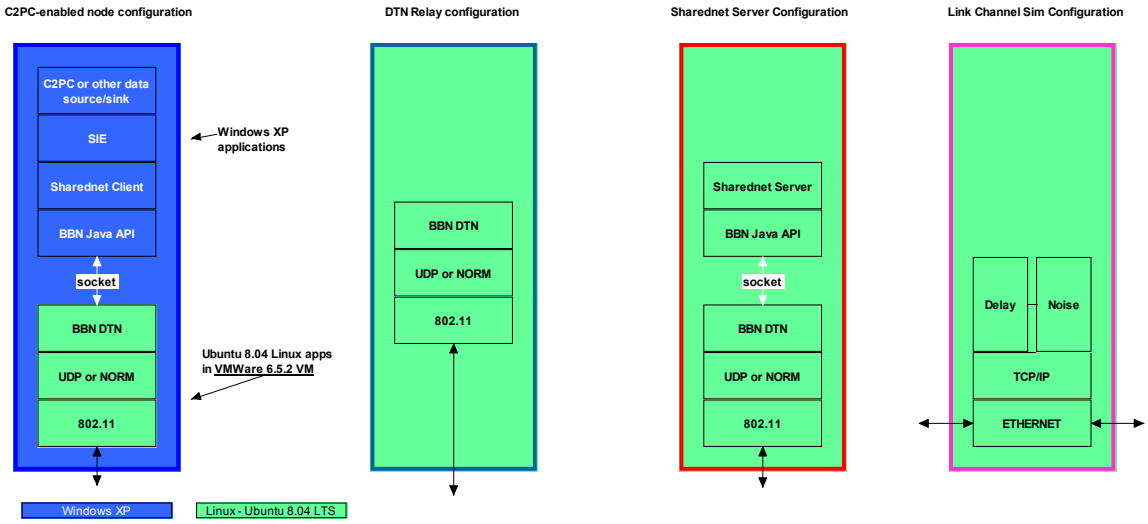


Figure 1: Computer configurations for system integration

The end-to-end lab topology was initially set up as follows:

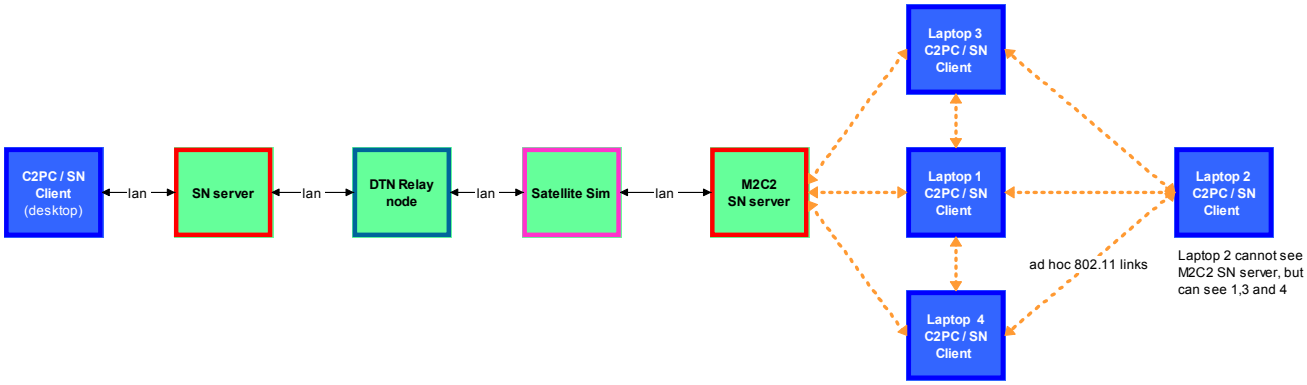


Figure 2: Lab topology for initial integration and MCBH SN test emulation

This network was set up with IP addresses and network functionality as required for the MCBH tests as detailed in section 3.2.2.3. (Not shown in Figures 1 & 2 are the commercial satellite emulators and Global Command and Control computer provided by SPAWAR for the MCTSSA testing.)

3.2.2.2 SharedNet – DTN Integration

Details of this task, which was done under separate DARPA contract with SPAWAR, are reported elsewhere. In summary, the modifications Jim McKelvey performed to the stock SIE/SN software to work with the BBN/MITRE-supplied included:

a) Incorporate BBN dtn-api library into build and run scripts.
b) Update SN to use Java 1.6 (required for dtn-api).
c) Write interface layer between SN and DTN, in particular to map IP addresses to EIDs (Entity IDs).
d) Write DTN Messenger to use interface layer instead of sockets.
e) Make changes to comm layer to support DTN.
f) Modify interface layer to support DTN multicast EIDs.
g) Modify interface layer/configuration to support aggregation.
h) Modify configuration editor to support DTN network types and EIDs.
i) Write log analyzers for node logs.
j) Modify configuration server/download to support DTN operation.

3.2.2.3 Integration Work and Associated Challenges:

The mundane details of the work required to put these topologies together and to do the system configuration and network management are not of particular interest for this report; suffice it to say that the configuration of over 20 computers with over 75 different IP addresses to configure and keep track of, and the attendant management of firewall, multicast and other system administration rules was a large part of the effort. Of particular interest however are some notable challenges we faced that need to be considered in future system design and integration activities of this nature:

3.2.2.3.1 Multiple Network Configurations

During the course of the project, the network IP addresses and the names of the network nodes were changed three times.

The initial integration setup had one set of IP addresses to mimic SPAWAR set-up. The nodes were called SSCPAC, CFT, B1386, Truck and Mobiles 1-3.

Then we found MCBH Marine Corps Experimentation Center (MEC) had another network to adapt to, using CIDR (Classless Inter-Domain Routing) masks to separate subnets from one-another. Some nodes were also renamed; SSCPAC became PANDA, and Truck became MIP.

Finally we had a 3rd topology for MCTSSA tests. Nodes were generalized to HHQ (Higher Headquarters), COC (Command Operations Center), TEP (Theater Entry Point), and then in the field, MIP became POP (Point-of-Presence)

For the MCTSSA testing, additional complexity was introduced with the addition of C2PC, the Global Command and Control System (GCCS) computer, and 2 extra satellite hops.

The net result was that the complexity of network management coupled with the need to change SIE, DTN and NORM configuration files from end-to-end underscores the need for additional future system engineering to make changes as transparent to operational deployments as possible.

3.2.2.3.2 Time Synchronization

Initially, the DTN implementation provided by BBN followed the RFC5050-compliant DTN bundle protocol requirement that DTN bundles have a bundle lifetime expressed in GMT; that is, a piece of information carried in a DTN bundle has a "shelf life" tied to a

specific time, after which any node may discard the bundle. This requires that all nodes in the DTN system be synchronized to GMT within some degree of accuracy.

The initial lab setup (to mimic the San Diego SPAWAR lab set up) made universal access to an NTP server problematic. We were using SPAWAR routable addresses in an isolated network, and therefore did not have access to an NTP server on the internet. We added a "backdoor" admin LAN to synch one test computer with a lab GPS Stratum 1 NTP server, then had the test node set up as a Stratum2 server for the other computers in the initial test topology. (This also aided post-test analysis by enabling time sync using time-stamped multicast packets on the administrative lan.)

While this solved some of the time-setting issues, two new issues were discovered:

1. The WindowsXP machines wouldn't always synch properly, and

2. The Linux VM clocks exhibited unacceptable drift characteristics, so that the Linux VM clocks kept diverging from the WinXP clocks (which themselves were constantly drifting from the NTP server time.)

Integration experiments were constantly failing due to clock drift, so eventually BBN provided a new Bundle Protocol Agent (BPA) that implemented a non-standard Relative Bundle Lifetime capability whereby the bundle expiration time was taken to be relative to the time of receipt of a bundle and not an absolute wall clock time. This solved the constant test failures due to unmanageable clock drift.

It was still desired to have system clocks in synch with GPS / GMT time to facilitate test data analysis, so in the lab, the strategy that was ultimately adopted was to synchronize all the Linux clocks (even VM guests) to a local time server, and then the WindowsXP clocks were then synchronized to their Linux Guest VM's.

We discovered that some of the problems synching Windows clocks were related to the fact that the default setting of the w32time.dll time-synch application is to synch the Windows clock to a time server once every seven days whether you need it or not. A registry modification was required to reduce the update period from a week to several minutes:

set the time synch interval (period) to something much less than the default of 7 days:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W32Time\Parameters

Period=<number of seconds between NTP server polls>

3.2.2.3.3 Addition of NORM and Windows Multicast Operation

In order to add some robustness to wireless links and to potentially cut down on traffic over the RF links, the Naval Research Lab NACK-Oriented Reliable Multicast (NORM) protocol was added below BPA, using a CLA provided by MITRE.

Addition of multicast provided a plethora of new issues to basic integration and operations with the WinXP/Linux VM setup:

1. SN now had to be configured to use multicast groups
2. Special Windows software development toolkit software (mcase.exe) had to be used to enable WinXP to join a multicast group. This also had to be run in a batch file in a continuous loop to get it to initialize and join the group without start up issues. For field deployment of this system, a much more elegant solution is needed.

3.2.2.3.4 Performance Tuning of Reliability Mechanisms

SharedNet, the BBN Disruptive Routing (DR) and it's mechanism of IP Neighbor Discovery (IPND) and NORM all have separate reliability mechanisms/timers and tuning them to work together properly took a large amount of time. While the final result was that the team got SIE/SN/BPA/NORM working well, much more tuning and operational configuration refinement needs to be done in the future.

3.2.2.3.5 Unexpected Satellite modem Quality of Service (QoS) Characteristics

During the MCBH field tests, the operational Ku-band satellite hop exhibited some unexpected jitter characteristics. While we anticipated a certain amount of jitter and delay, the operational system would delay packets over the satellite link for sometimes seconds at a time due to the queuing behavior on the satellite link. This made performance tuning of the aforementioned reliability mechanisms problematic. This behavior was later attributed to a misconfiguration of the modem at MCBH, but pointed out the need to consider such unexpected behavior in future systems design of systems which rely on link layer ARQ retransmission mechanisms.

(The feasibility of modeling simulating such large delays and jitter measured in seconds was discussed with the engineers at Apposite. They explained that every satellite modem manufacturer has their own proprietary way of doing QoS, and that most modem makers are unwilling to share their algorithms, therefore it is near impossible to model these effects.)

3.2.2.3.6 SharedNet Analysis Tools

The SharedNet software in the proper debug mode creates voluminous logs that show every detail if the operation of the software. In order to capture statistics and to quantify

the operation of SharedNet with and without DTN, it was necessary to do a considerable amount of post-processing of the logs. Team member Rick Borgen created a number of post-processing programs that dealt with these SN log files as described below. (The actual scripts (php) are available for download on the BBN SIE Wiki.)

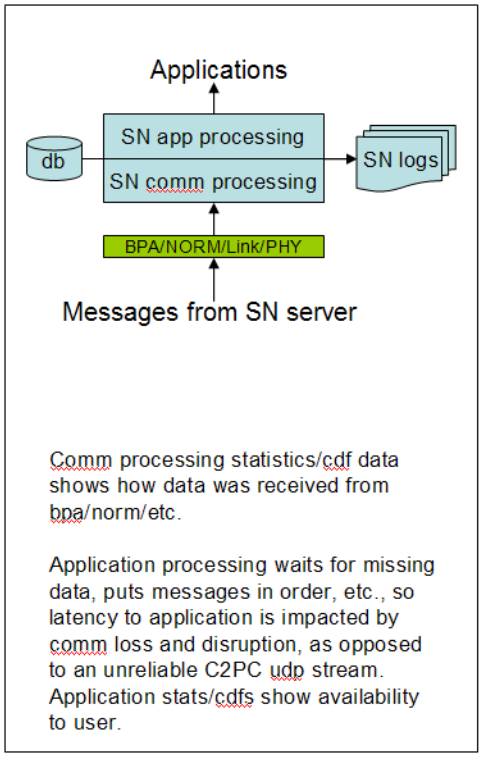


Figure 3: SharedNet Log processing

The most important source of test metrics are the SharedNet logs provided normally by the SharedNet system (Figure 3). These logs provide highly detailed information about transactions at various stages of processing for each running ShareNet application. However, the complexity and format of the logs offers considerable challenges to support analysis of the test results. Just one test run of SharedNet will create dozens of very large files packed with complex data with much of this data only suitable for specialized trouble-shooting. From a test analysis point-of-view, the problems included far too much volume, formats poorly suited for direct reading, too much irrelevant information, inconvenient formats for automated processing and no summarization or statistics. The solution was to develop a set of tools for filtering, reformatting and deriving useful data from the logs, with further tools for preparing test-based reports and plots.

The first stage of SharedNet log processing aims to capture only the most relevant data, format that data more conveniently and compute useful statistics. The main object of interest is the single transaction event as seen by the communications layer and then as seen at the application layer. Other objects of interest are events related to the detection and correction of missing messages. Format simplification included reducing the

essential information to a single line of text, providing human-readable date-time formats and using a fixed format. The tools also compute various conventional statistics such as totals, averages, and min/max, as well as some special observations such as data gaps. There are three standard products from first stage processing:

events – one-line time-tagged descriptions of key events

intervals – summaries data flows/gaps over regular intervals

stats – statistical summary of total test run

The second stage of SharedNet log processing operates on the first-stage products (usually stats and intervals) producing reports and plots suitable for analysis. In general, this immediate next-stage product is in the form of a comma-separated-values (csv) file that can be readily imported into Microsoft Excel. An important part of this process uses a simple database of test descriptions to attach the test identification and test context information with each reported metric. For example, the most general report tool summarizes selected statistics in row/column style with each line describing a specific test run along with its statistics. Additionally, there are two specific analysis products used principally for plots:

cdf – cumulative distribution function showing aggregate messages by time

aging – an aging analysis of track information over the lifetime of the test intended to provide a composite metric of the database quality

Post-processed data from the analysis software was posted on the BBN SIE Wiki site for use by the other teams who were doing the detailed analysis of the test results. Some examples of the data thus derived are shown in the Test Results Summary section below.

3.2.3 Test Program

This section will summarize some of the JPL Core Engineering Support team's observations of the field and lab tests, which were planned and directed by SPAWAR. This section will summarize the results with a focus on the benefits DTN adds to standard SIE/SN. During this phase of the program, BBN was named the lead integrator, with SPAWAR being in charge of the testing, so the details of how well SN Classic, SN/DTN and C2PC worked will be reported on by BBN and SPAWAR in separate reports.

Field testing in Hawaii was conducted in two phases; a dry run in December 2009 (one week at MCBH), followed by the formal field tests conducted from 19 January 2010 through 29 January 2010. Following the work at MCBH, we prepared the Protocol Technology Lab at JPL to host the MCTSSA testing activity, which was conducted between 22 February 2010 and 25 February 2010.

3.2.3.1 SIE/SN MCBH Field Tests

The field testing was conducted over a 2 week period in January 2010. Over 125 test runs were completed, and over 2,000 SharedNet log files were captured and processed through the SharedNet analysis scripts described in section 3.2.2.3.6 above, and plotted in Excel.

During the field testing at USMC Base Hawaii, the performance of SIE/SharedNet without DTN (referred to as SharedNet Classic) and SIE/SharedNet over DTN (BPA/DR/IPND/NORM) was evaluated. As a general rule, the observed results were as expected, with DTN enabling the delivery of data to intermittently connected mobile nodes that the standard internet-based SIE/SN system could not reach.

3.2.3.2 Topology – Marine Corps Base Hawaii

The SPAWAR Test Plan [1] for the field tests called for an overall test scheme whereby SIE data was generated in their lab in San Diego, passed through the DISN network to Hawaii, where, after a satellite hop to their M2C2 Interoperability Prototype (MIP) vehicle, the SIE data was transmitted to laptops in the field over Secnet11 (a secure 802.11 system.) The planned topology is shown below:

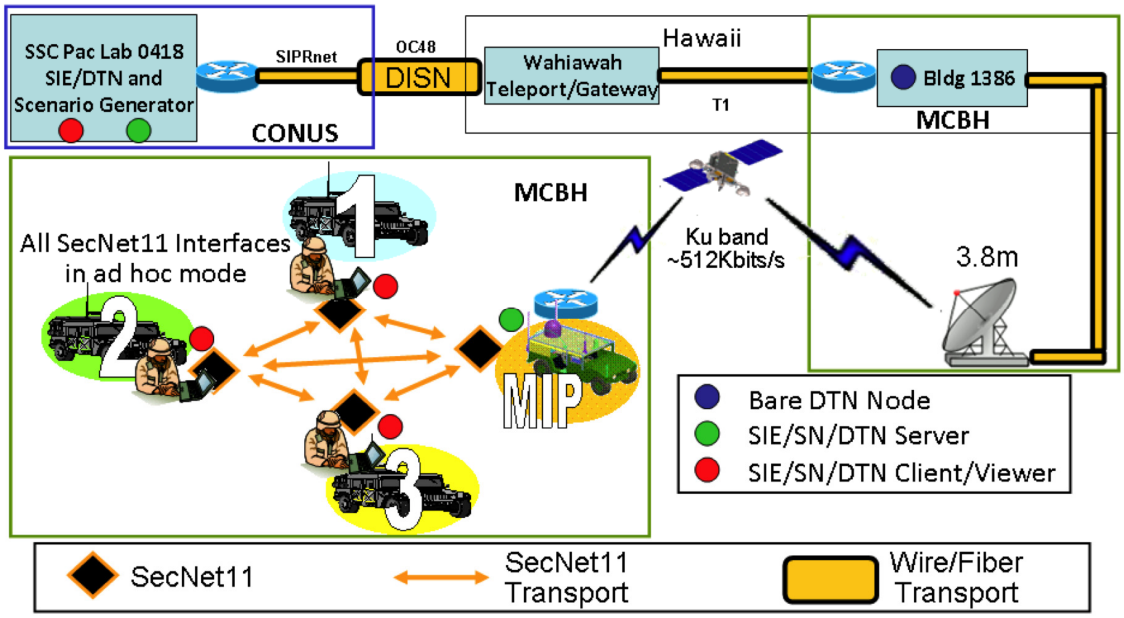


Figure 4: MCBH Field Test Topology

In order to duplicate this setup in the lab at JPL for integration and test before going to the field, the MCBH network as shown in figures 4 and 5 was functionally duplicated in the PTL. A WANEM simulation of the Ku-band satellite hop was used to simulate the delay and losses expected in the field tests. 802.11 RF links were used in place of the SECNET 11 cards planned for the field tests. The source of track data was the SPAWAR DARS Replay software that can

record and play back SIE/SN track data. As explained in the SPAWAR Test Plan, "The "Scenario Generator" listed in the SSC Pacific block of the test architecture is a specialized component of the SIE called the Data Archive and Replay System (DARS). DARS can record SharedNet events and replay them at real or accelerated time. SN objects (tracks, overlays, etc.) are generated over a specific time period and then replayed for each test to ensure consistent input."

(Field tests used NIPR net, with SECNET11 cards)

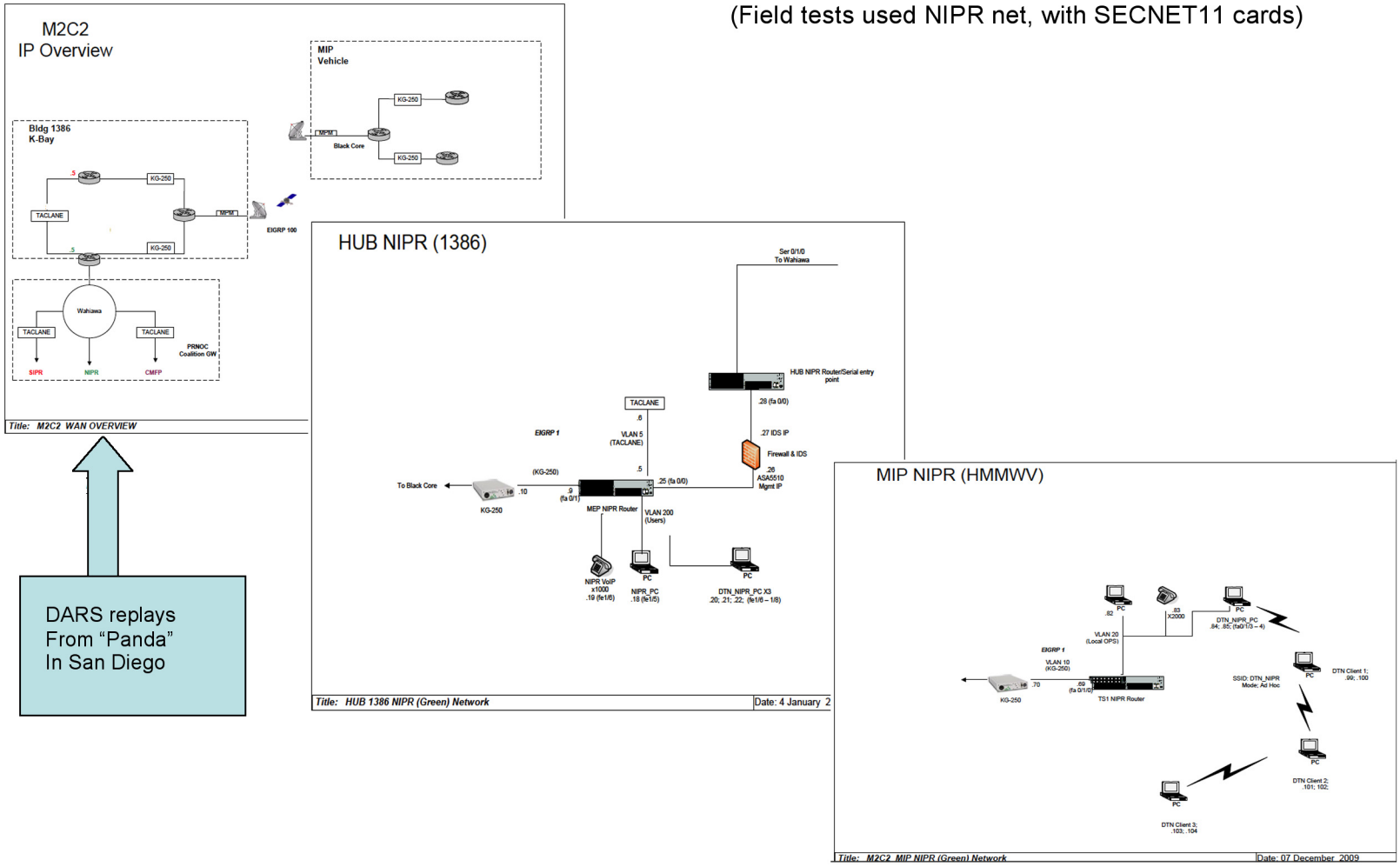


Figure 5: MCBH network functionally duplicated in JPL PTL

3.2.3.2.1 Test descriptions and scenarios

The overall topology and configuration details for the MCBH field tests were detailed in figures 4 and 5 above. The DARS replay system was used to provide repeatable and consistent track/overlay generation data flows for transmission over the various links.

The test configurations and mobile topologies used are detailed in the SPAWAR Test Plan [1], and summarized here in figures 6 and 7 following, which excerpts drawings from the SPAWAR test plan.

The first 5 test cases were merely to characterize the performance of each of the communication paths between each node. A utility program called MGEN* was used to present a known stream of traffic to the network, and to characterize the throughput, latency, jitter and loss over each link.

The subsequent test cases (6-13) were designed to test the end-to-end performance of SIE/SN in both "classic" UDP/IP-based mode, and over DTN, with various mobile scenarios played out that would represent potential tactical topologies of interest to the USMC.

While most of the tests compared SIE/SN/UDP/IP ("classic" mode) versus SIE/SN/DTN, an additional multicast forwarding test was run when the test topology called for a linear topology of mobile units. In a linear topology of RF links (see test case 9, figure 6), data from standard SIE/SN/UDP/IP would not be able to reach mobile2 or 3 from MIP, as there is no standard internet route from MIP past mobile 1. While DTN, with its store-and-forward capability, can overcome this deficiency, an experimental alternative method of doing multicast forwarding was tried by BBN. While it succeeded in forwarding some of the multicast data from MIP through mobile 1 to the other mobile units, it's net performance was orders of magnitude worse than that of DTN.

A typical SIE/SN "classic" versus SIE/SN/DTN test run is shown in figure 8, and described in the next section.

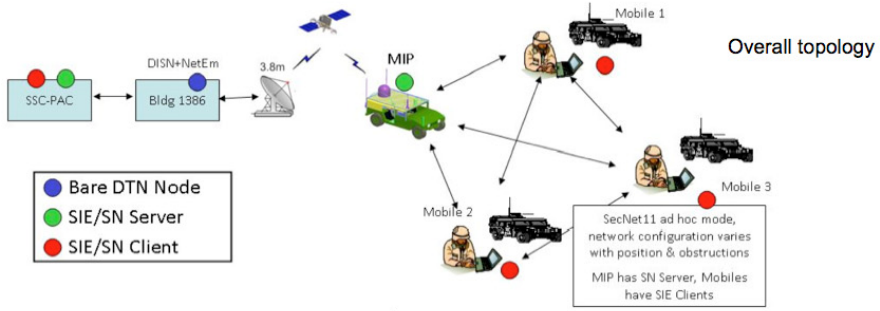
---

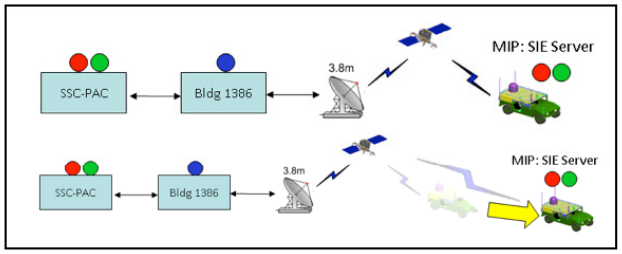
*From http://cs.itd.nrl.navy.mil/work/mgen/ -- "The Multi-Generator (MGEN) is open source software developed by the Naval Research Laboratory (NRL) PROTocol Engineering Advanced Networking (PROTEAN) Research Group. MGEN provides the ability to perform IP network performance tests and measurements using UDP/IP traffic."

- Test Cases 1-5: Baseline link characterization of DISN, SATCOM and SECNET11 using MGEN
- Cases 6-13 : Performance Tests:

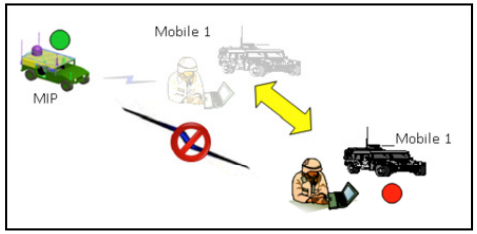
  SIE/SN "classic" versus

  SIE/SN/DTN

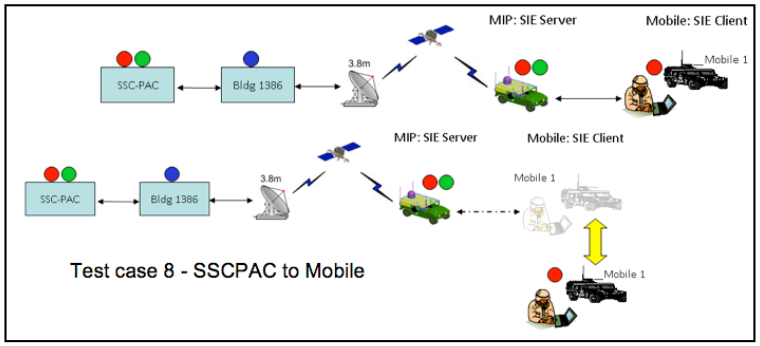




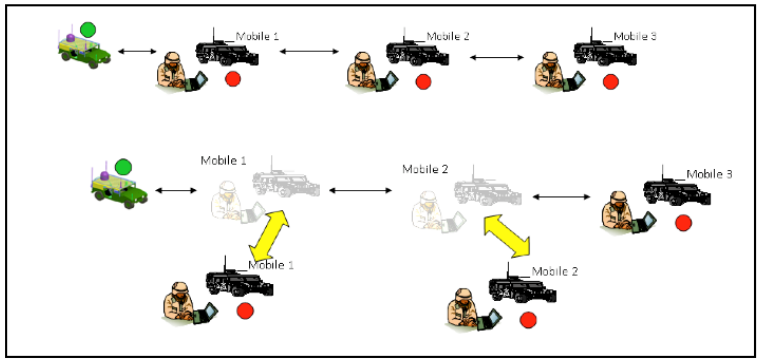
Test case 6 - SSCPAC to MIP



Test case 7 - Wireless - one laptop

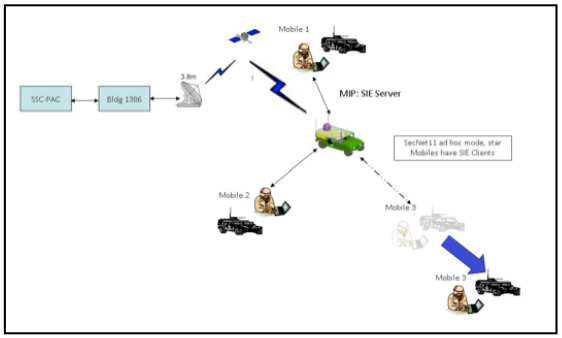


Source: SPAWAR SIEDTN Test Plan v1.1, January 13, 2010

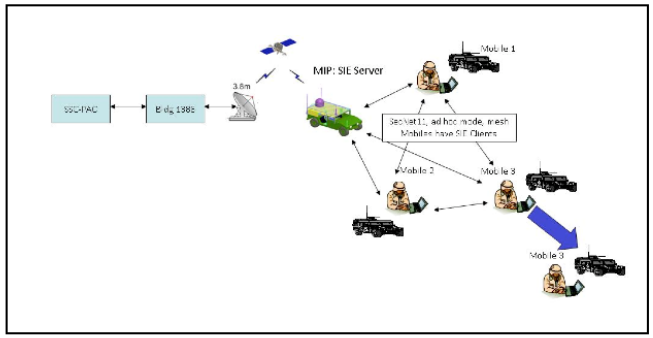


Test cases 9a,b - MIP to 3 Mobiles, ATH, OTM

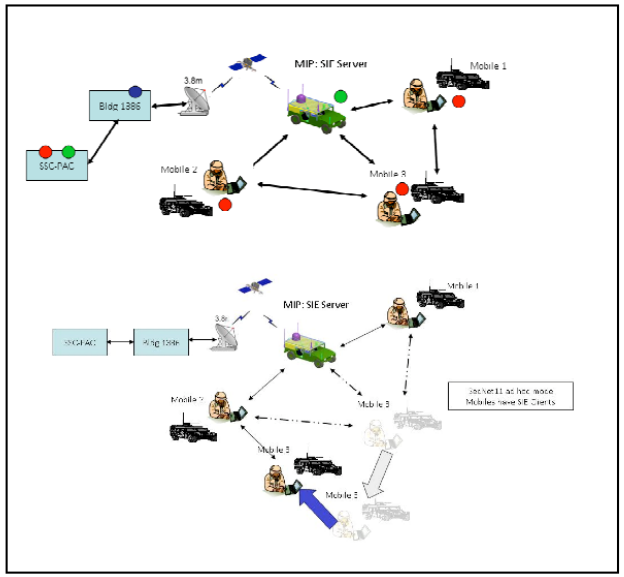
Figure 6 – MCBH Test Cases 1-9



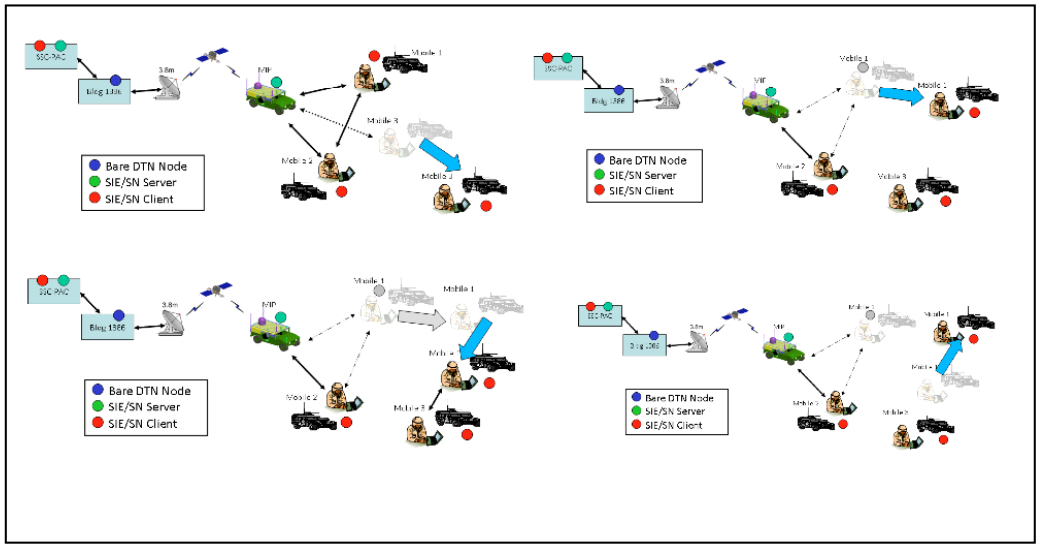
Test case 10 - SSCPAC to Mobile Star



Test case 11 - SSCPAC to Mobile Mesh, 1 mobile leaving



Test case 12 - SSCPAC to Mobile Mesh, mobile leaving & reconnecting



Test case 13 - SSCPAC to Mobiles with Data Mule Operation

Figure 7 – MCBH Test Cases 10-13

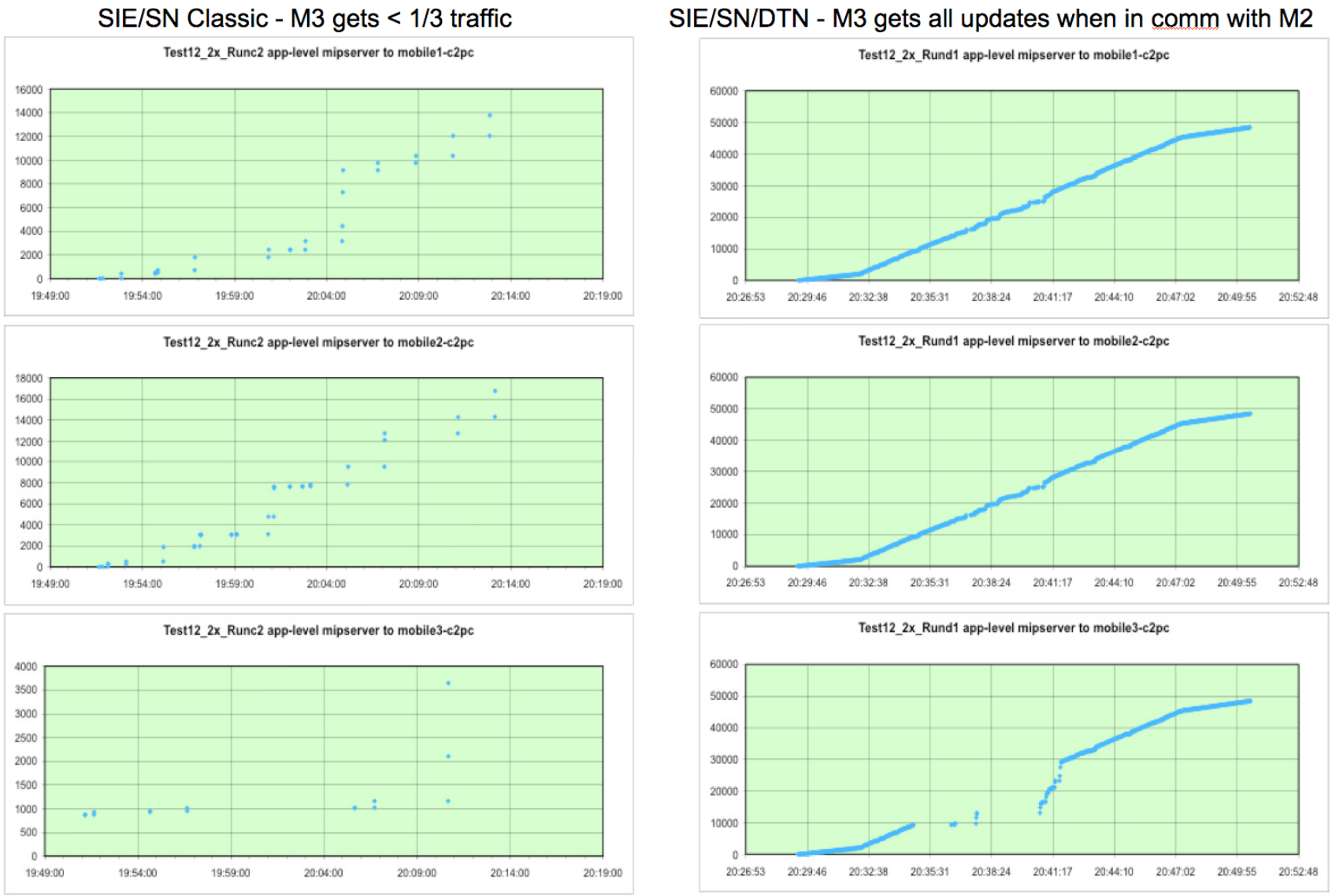


Figure 8 – Test Case 12 Example Runs

3.2.3.2.2 MCBH Test Summary and Example

Figure 8 shows cumulative track update counts at each node for a run using test case 12 topology. The plots on the left hand side are the data received at the 3 mobile laptops over the 802.11 links using the "classic" SIE/SN system that uses normal wired internet protocols. Out of close to 50,000 track updates during the test, the plots show that the mobile units received less than one third of the data due to the breakdown of standard internet protocols and their need for continuous end-to-end connectivity to be successful.

On the right hand side are plots from the three mobile units when the SIE/SN system was run over DTN protocols. In each case, all mobile nodes received all the track updates that were sent to them by the MIP. During this scenario, mobile 3 moved out of communications range for several moments and lost contact with the MIP, causing disruption in the track update flow. (See bottom left plot; gaps in data updates between about 20:35 and 20:41.)t Mobile 3 then moved close to mobile 2, and as soon as communications was established with mobile 2, the DTN protocol allowed stored update data addressed to mobile 3 to flow, resulting in a rapid "catch-up" in track updates (steep slope around 20:41.) After the "catch-up", mobile 3 continues to receive updates at the rate MIP generated them, but in this part of the scenario, mobile 3 is receiving them through the DTN store & forward capability from MIP through mobile 2 to mobile 3.

Out of the hundred-plus test runs, this pattern was demonstrated over and over: SIE/SN in its classic wired internet incarnation performs poorly in a wireless or satellite hop environment, while SIE/SN running over DTN consistently delivered the expected and desired amount of data to all end nodes in on the disconnected edge of the network.

There were issues discovered (yet to be resolved) with the performance of classic SIE/SN, and there is much to be learned about performance tuning from the enormous quantities of data that were taken, and those details will be discussed in reports from SPAWAR and BBN. Suffice it to say, the MCBH tests demonstrated the value of DTN in a large number of scenarios over an operational USMC satellite communications network.

3.2.3.3 SIE/SN MCTSSA Testing –

Following the successful field demonstrations and tests in Hawaii, the original plan was to turn the system over to the Marine Corps Tactical Systems Support Activity (MCTSSA) at Camp Pendleton, California for independent test and evaluation of the DTN-enabled SIE system.

Scheduling conflicts and programmatic timing made the tests at MCTSSA impossible, so it was decided that the MCTSSA tests would be conducted at JPL under the supervision

of Capt. John Frushour, the Chief Engineer for Marine Corps Tactical Networks, with observers from the US Army CERDEC activity.

Lab testing was conducted over a period of four days, with 22 test runs of 20 minutes each were conducted under controlled and repeatable circumstances. This allowed the confirmation of performance seen in the field tests, expanded the operational envelope to include two satellite hops (with much worse satellite hop performance than seen in the field tests,) used a currently fielded track generator (Global Command and Control System, GCCS), and showed the performance of the C2PC program in parallel with SIE/SN/DTN.

3.2.3.4 Topology – MCTSSA tests

The MCTSSA topology was different from the arrangement used at MCBH in terms of topology, systems used for data (track) generation, and comparison tests.

It was decided that we would emulate a tactical network which included two satellite hops, each with a random 65% availability, and that we would test SIE/DTN against a currently-fielded Northrop-Grumman application known as C2PC.

To add both rigor and realism to the tests, a network was set up that included the Global Command and Control System (GCCS), which generated track data provided to both C2PC and the SIE/DTN/NORM system tested in the field. The topology was set up to allow simultaneous simulation scenarios to be run in parallel for direct comparison of the current system end-to-end IP-based system (C2PC) and the DTN-enabled system (SIE/SN/DTN/NORM).

At DARPA's request, two Apposite Linktrophy link simulators were borrowed and included in the simulation. Their performance was similar to the WANEM simulators used, but they had an advantage of providing graphical real-time monitoring of the performance of the satellite links during the tests.

The next two figures provide details on the configurations used in the PTL for the MCTSSA testing: Note the changes in node names to more generic identifications.

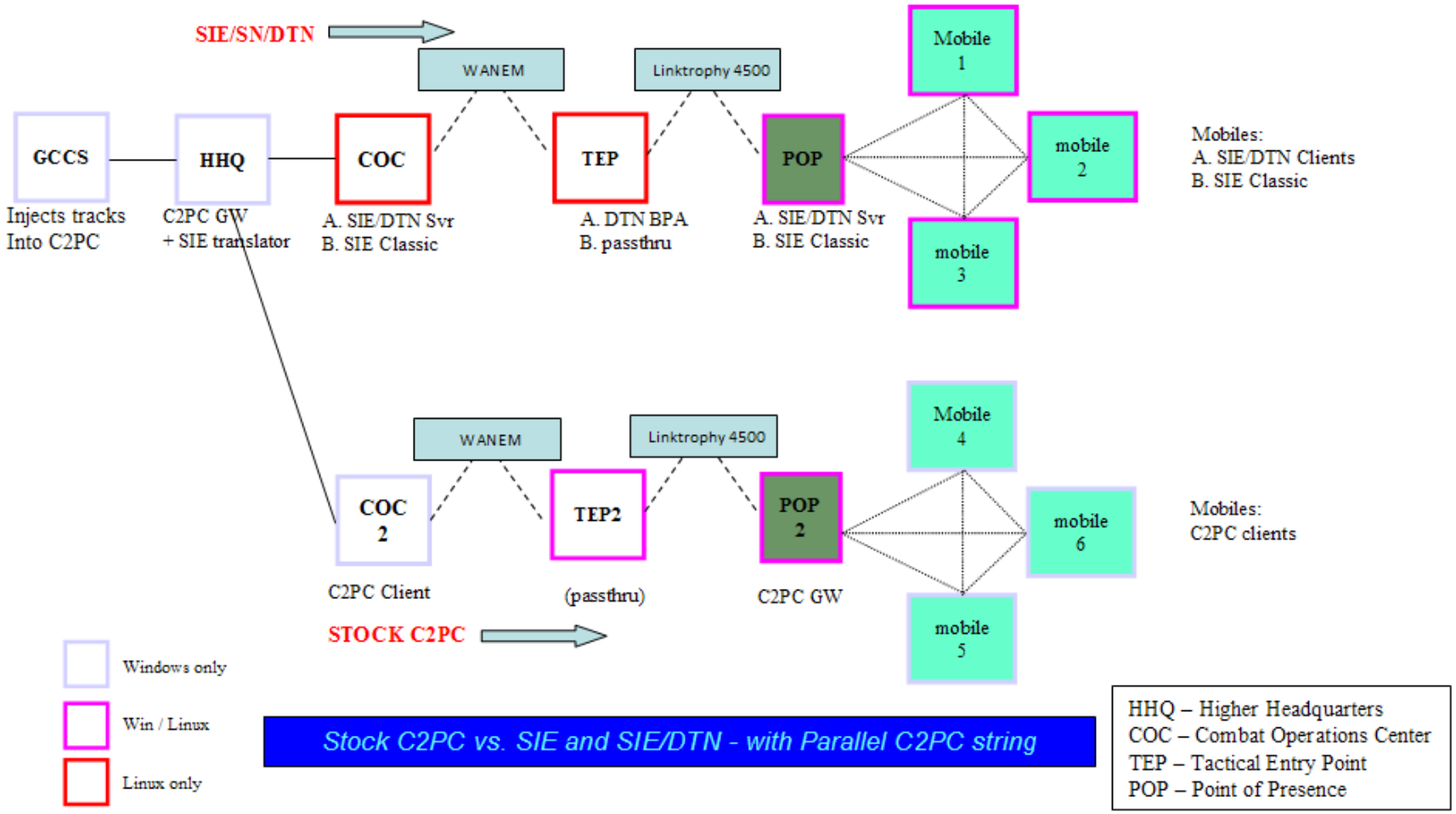


Figure 9: MCTSSA Test Topology

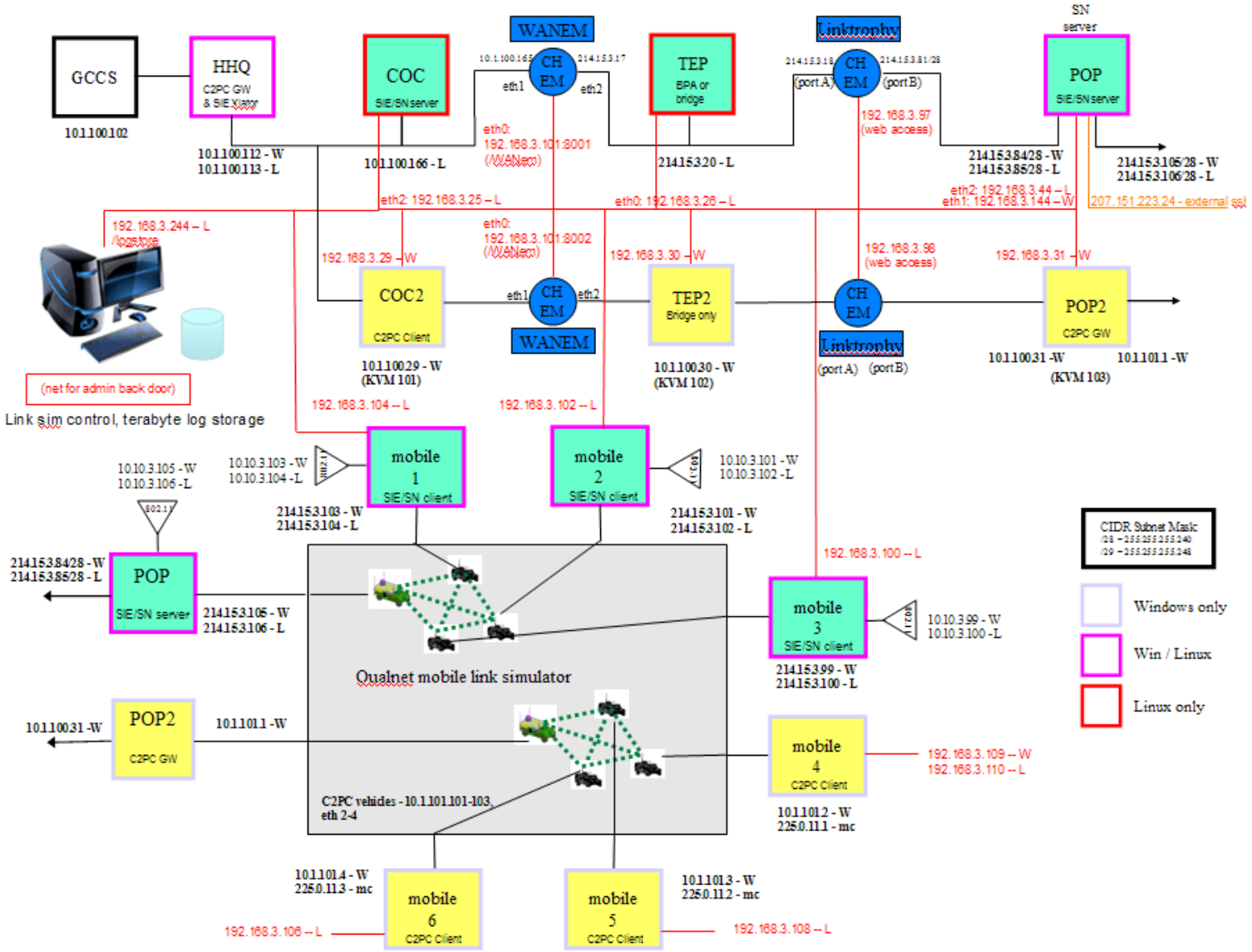


Figure 10: MCTSSA configuration details

3.2.3.4.1 MCTSSA Test Descriptions and Scenarios

The satellite hop used in the MCBH tests was replaced by two satcom hops in the MCTSSA testing. Commercial Apposite Linktrophy 4500 network simulators were used for one pair of satellite hops, with WANEM simulations used for the other pair of satellite hops.

The satellite availability in the MCBH tests was very good, but for the MCTSSA testing it was desired to stress the system and use a rather worst case satcom link. During the DARPA Phase 2 tests at Fort A.P. Hill, the overall availability of the satellite channel was about 65%, as shown in figure 11:

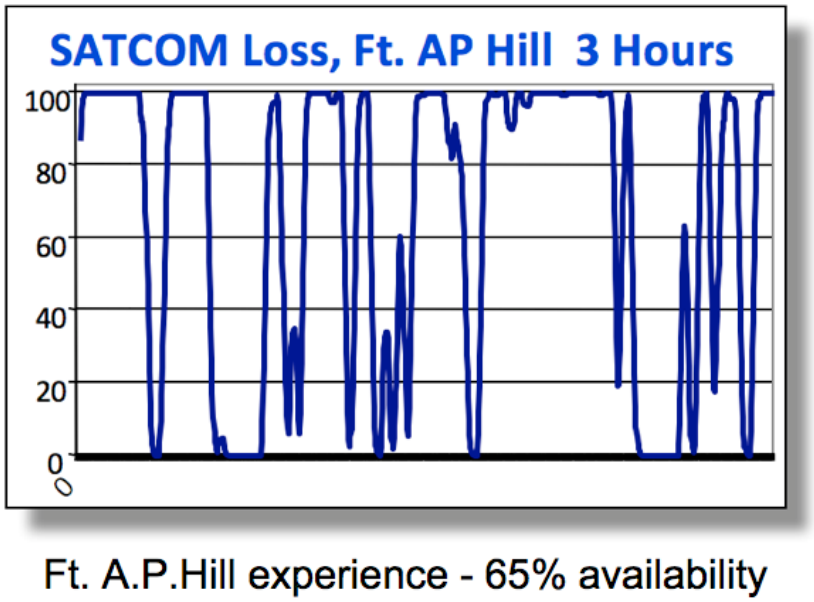


Figure 11 – Ft. A. P. Hill tests

By request of MCTSSA, both satellite link simulations were programmed to provide a 65% availability over a 15 minute period, after which a period of 5 minutes of good connectivity was simulated. A random pattern of outage was independently chosen for each of the two satellite links that provided an aggregate 65% availability over each link (figures 12-13), and the overall throughput used for the satellite links is shown in figure 14. When it was noted how bad the resultant throughput was, it was decided to leave it as programmed as it would show how well C2PC and SN/SIE/DTN would fare in pretty stressful conditions.

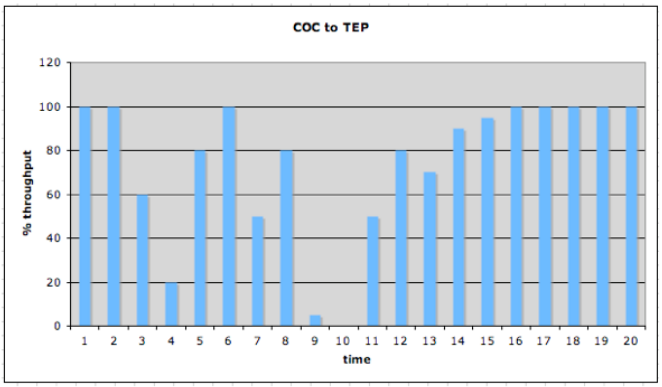


Figure 12: WanEm Simulator programming

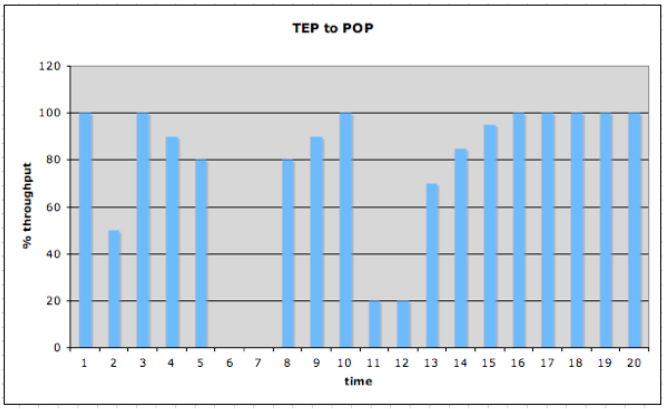


Figure 13: Linktrophy 4500 Simulator programming

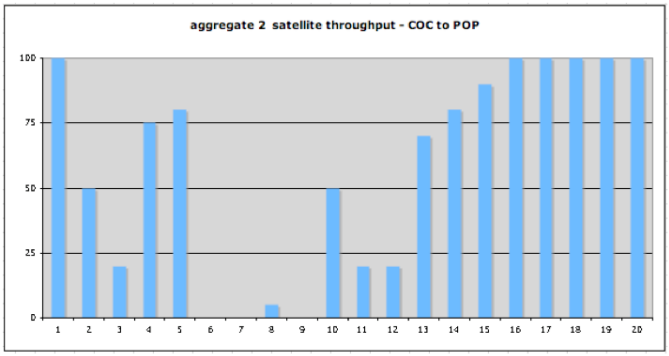
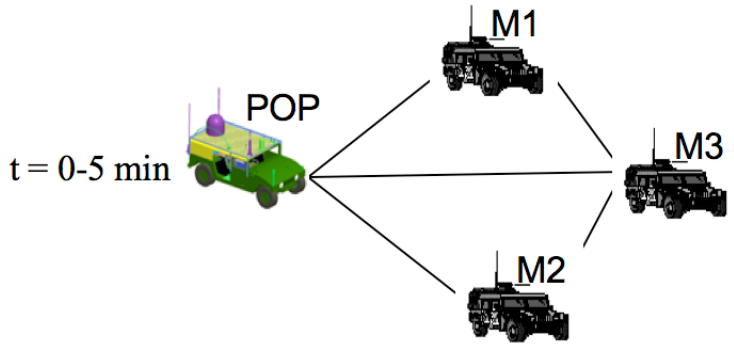
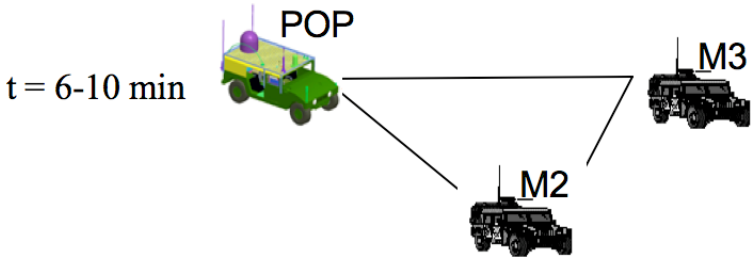


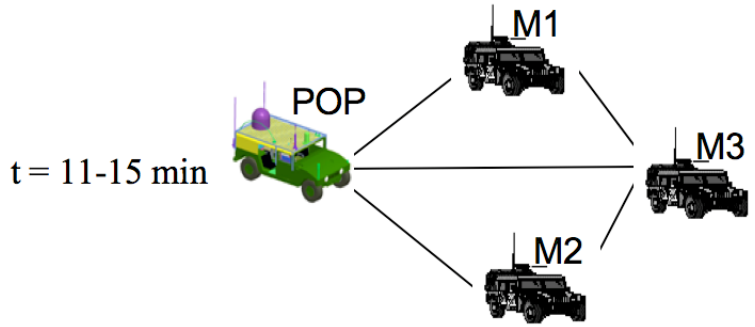
Figure 14: Overall effective throughput from COC through 2 satellite hops to POP

The test plan for the MCTSSA Tests (reference [2]) was prepared by SPAWAR, and 6 scenarios were prepared wherein three mobile units were moved in various tactical formations. In the interest of time, it was decided that only 5 mobile scenarios would be used, and they are presented in figures 15-19 below.



MCTSSA Scenario 1 :
Mobile 1 moves away and loses contact with POP, then gets back in comm





OBSERVED: SN/DTN brings contacts to M1 faster than retransmit requests up to SN server because he connects to M3 before he connects to POP

Figure 15: MCTSSA Scenario 1

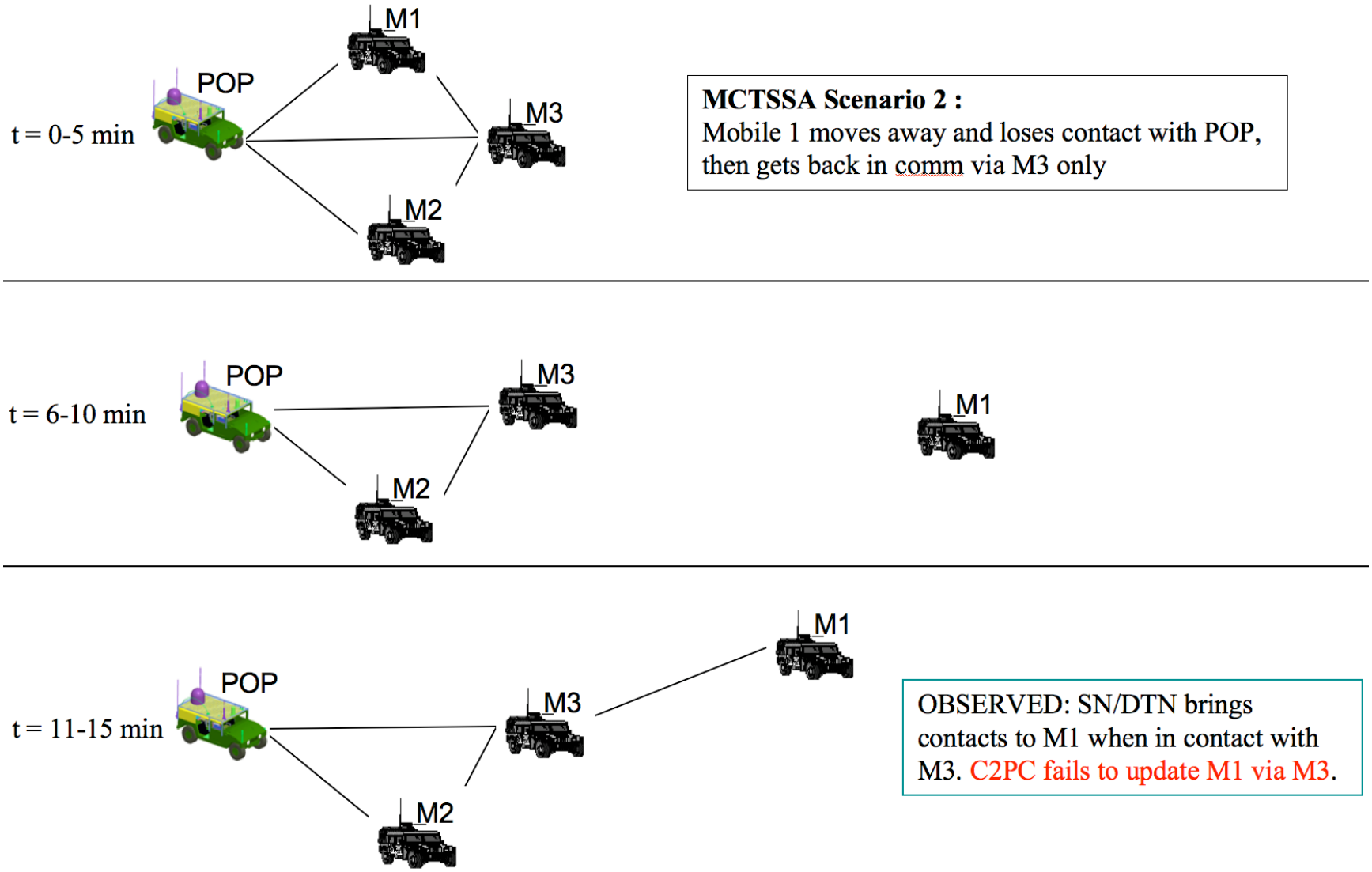
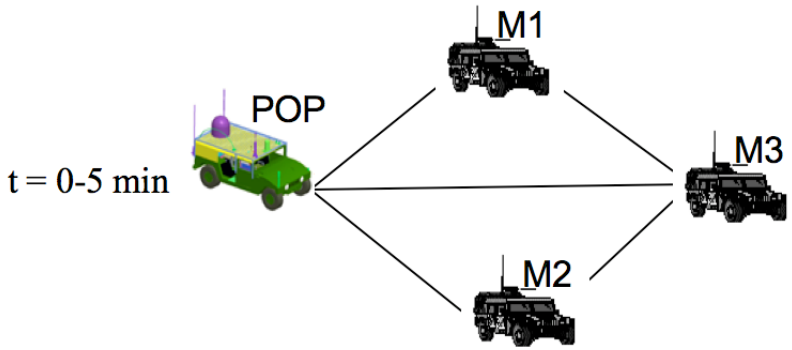
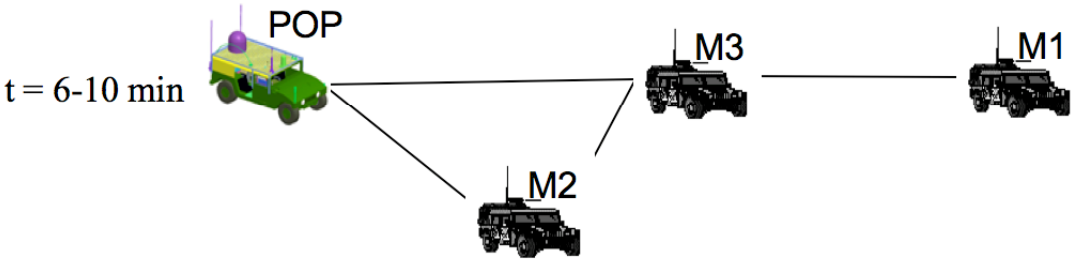


Figure 16: MCTSSA Scenario 2



MCTSSA Scenario 3 :
Mobile 1 and later Mobile 3 moves away from POP and forms linear topology



Observed: SN/DTN gets all contacts to M1 and M3.. C2PC fails to update M1 after t=8, and looses M3 after t=11

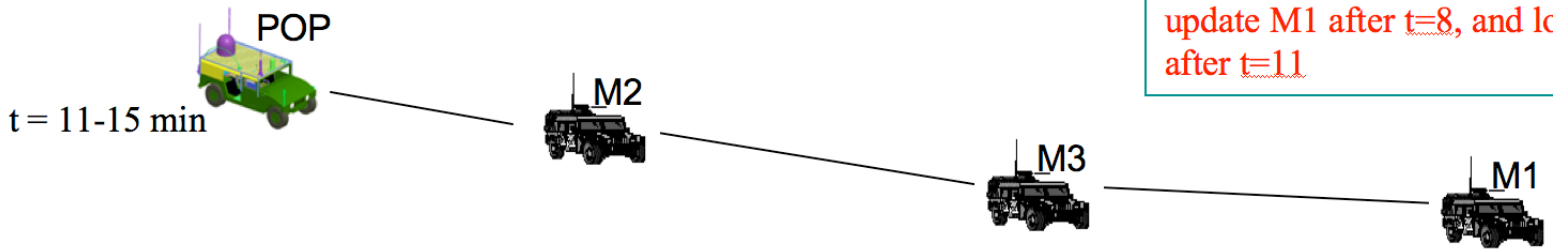


Figure 17: MCTSSA Scenario 3

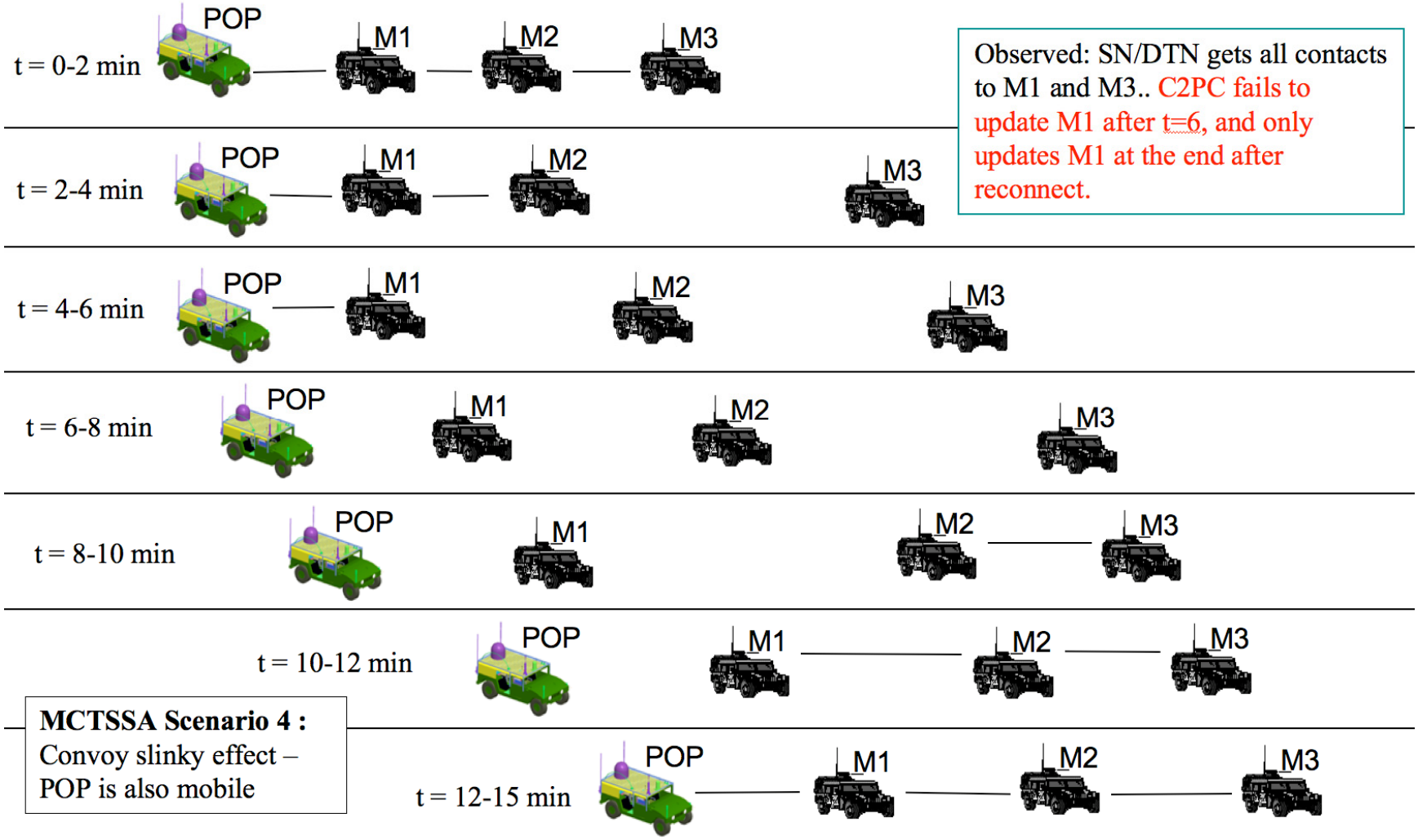


Figure 18: MCTSSA Scenario 4

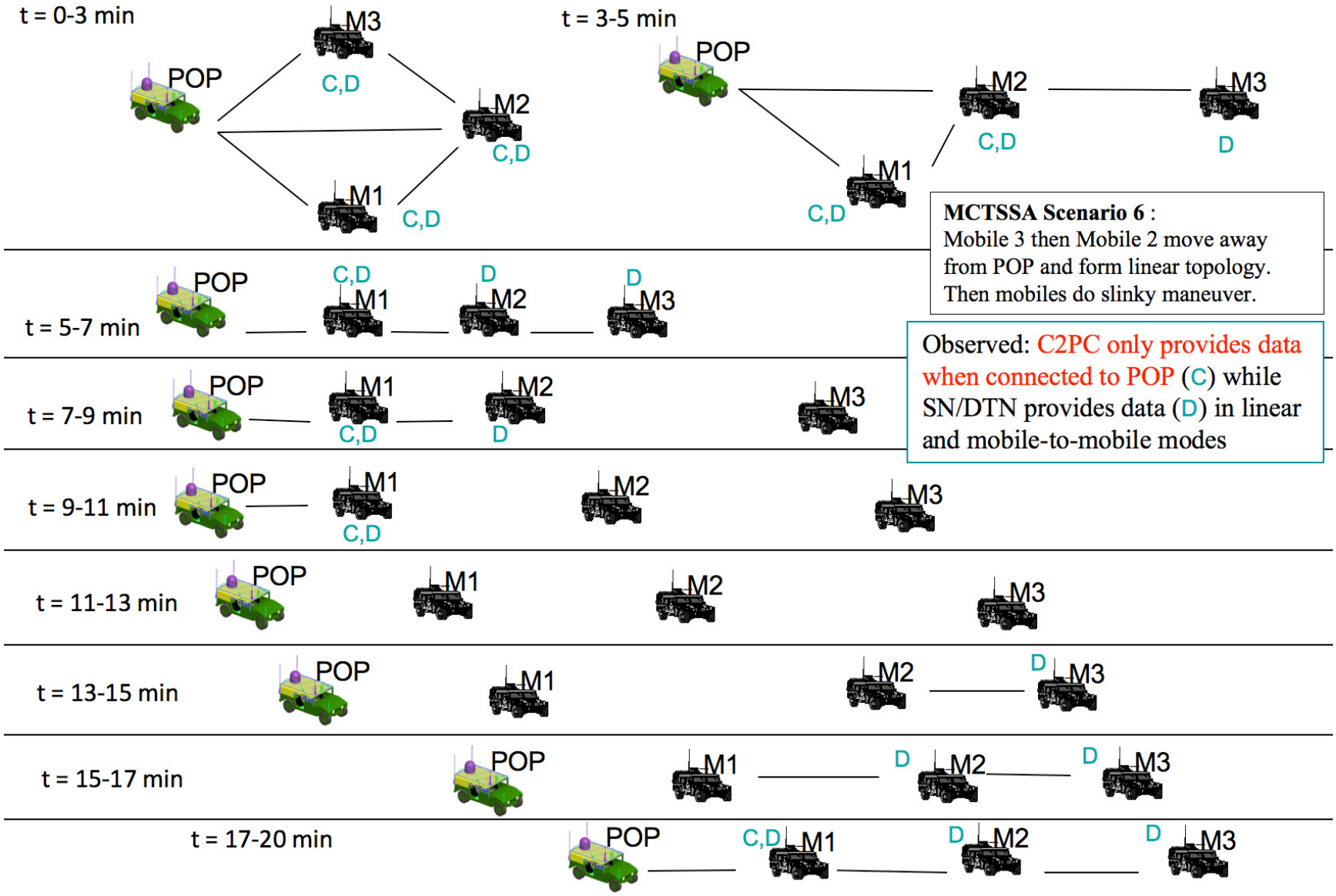


Figure 19: MCTSSA Scenario 6

In the lab tests, actual mobile laptops were used for the mobile nodes, but the 802.11 (SECNET11) RF links and vehicle mobility were simulated using a commercial network simulation software package named Qualnet. The Qualnet simulations used certified 802.11 protocol models, realistic link parameters (e.g. physical layer RF parameters), and moved the mobile nodes according to the planned scenarios. This resulted in a high-fidelity and perfectly repeatable series of tests that took much uncertainty out of the test-to-test comparisons of the different higher-layer protocols.

The MCTSSA tests followed a general procedure of starting up all software on all nodes, insuring that all DTN nodes were properly communicating over perfect links, then starting the satellite loss model scripts and the Qualnet mobility scripts. The mobility runs were generally 15 minutes long, with a five minute period of good communications at the end for evaluation of both DTN and C2PC loss-correction mechanisms.

Figure 20 shows a typical Qualnet screen shot during a simulation run.

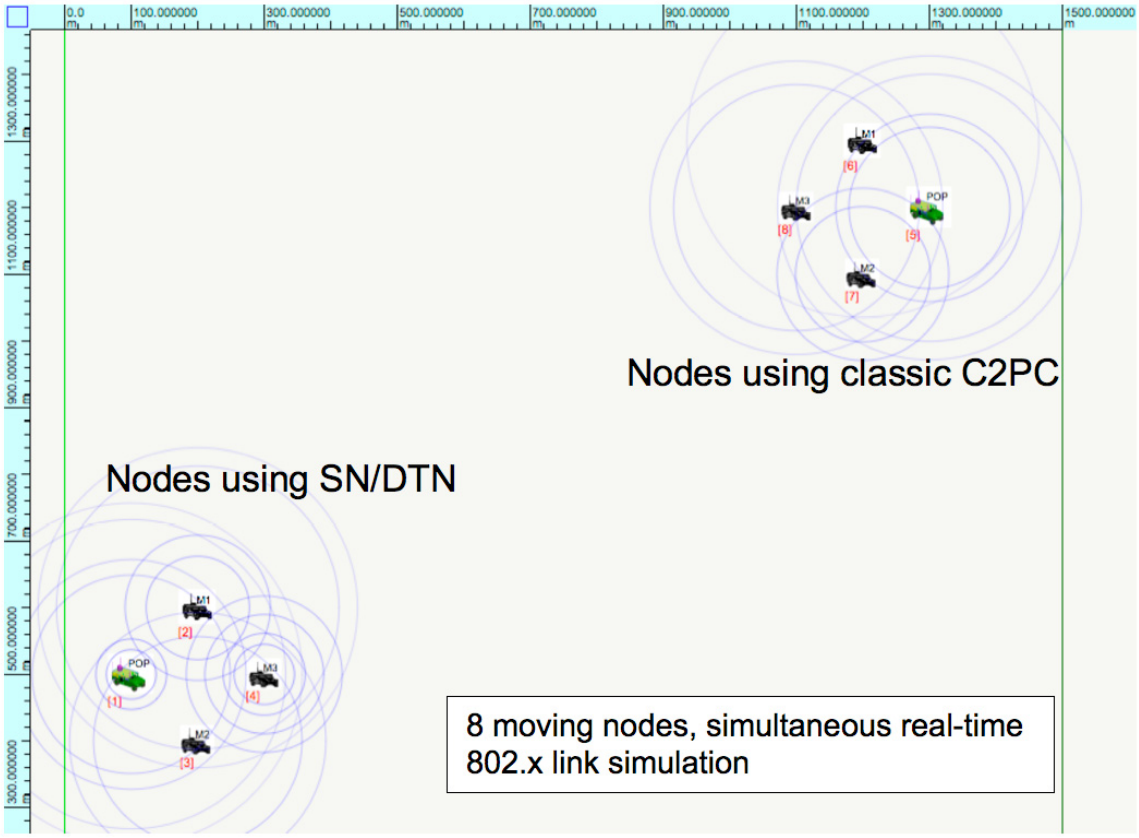


Figure 20: Qualnet Simulator screenshot

The simulator shows vehicle movement, lines between vehicles to indicate connectivity, and concentric expanding circles indicating packet propagation from each node. The details of the Qualnet simulation and the setups needed for these tests are included in Appendix C.

3.2.3.4.2 Configuration Changes from MCBH Testing

Aside from using Qualnet and satellite simulators to replace field equipment, there were other noteworthy configuration changes in the MCTSSA testing as compared to the MCBH field testing:

A Sun SPARC workstation with the Global Command and Control System (GCCS) software was used to generate track data and updates instead of the DARS Replay software. The GCCS box was used to generate tracks for both C2PC and the DTN-enabled SIE/SN system using a SPAWAR-developed translator program that converted GCCS/C2PC tracks/updates into SIE/SN tracks/updates.

An administrative Ethernet LAN was added to the lab configuration, and an administrative server with a terabyte disk reserved for test data was added. This enabled the automation of data collection (logs, etc.) before, during and after the test runs. The admin server (and its data) were made available over the internet to SPAWAR, BBN and MITRE for post-test data analysis. This saved a considerable amount of time compared to the manual process of copying data off each node post test onto a portable drive, as was done in Hawaii.

3.2.3.4.3 Emulation and Instrumentation Challenges

As was mentioned in section 3.2.2.3.5, the queuing behavior of the satellite modem at MCBH was unusual, and may have been exacerbated by misconfiguration. In any case, the latency behavior of the satellite modem in the field was not duplicated nor simulated in the MCTSSA tests. It was felt that the impacts of the large latencies were understood and not crucial to investigate in the MCTSSA testing.

As it turned out, an unexpected challenge was that the C2PC system neither provides logs nor visual indication (counts) of the number of track or track updates received. Furthermore, the inspection of C2PC data using Wireshark data was somewhat futile, as we had no way to correlate tracks between C2PC and the DTN branches of the experiment. By observing the track updates on the C2PC screens, it was very obvious when the C2PC system ceased to update, while the DTN systems were still getting track updates. This made it impossible (as of this writing) to quantitatively compare SIE/SN/DTN, though the disruption-tolerance of the latter system was obvious.

To compound the difficulties the data analysis team faced post test, the GCCS system did not have any way to script a test run with an exact number and distribution of tracks and track updates. A command line debug utility was available to create tracks at a certain rate, but there was not way to script it to provide a reproducible simulation.

JPL's Phil Tsao developed a parser for Classic SharedNet that could pull out track identification out of SN data flows captured by Wireshark, and this was provided to DARPA for use in subsequent analysis work.

3.2.3.4.4 Test Summary – Two Examples

Figure 21 shows the cumulative track update data plots from a run of scenario 1. The first plot shows the data from the GCCS system sent over the dual satellite link path to the POP SIE/SN server. The next plot is the overall link throughput, with the data received at POP in the bottom plot. Blue call-out notes are as follows:

1. Clean translator-to-COC1 events (top left plot) disrupted by satellite outages - gaps in POP1 data correspond to low satellite channel availability.
2. POP1 still receives all of the generated track updates as DTN forwards them when satcom link improves.
3. During a poor satcom coverage period, mobile2 and mobile3 receive DTN updates - mobile1 has by this time moved out of comm with anyone, so
4. Mobile1 finally gets caught up to the note 3-level of updates when back in range of mobile3 and the mesh network.
5. From point 5 on, all 3 mobiles get updates from POP1, ending up with all of the GCCS-generated track updates.

This demonstrates DTN's ability to forward data from mobile3 to mobile1 (where classic SIE/SN/IP would only be able to forward data from POP to mobile 3 at a later time), and the store-and-forward operations of DTN improving operations over the poor satellite path. In a classis SN/SIE or C2PC system, when contact is lost over the satellite system, the SN or C2PC servers have to go through all the handshaking necessary to re-establish the server-client relationship between the COC server and the POP server, which proved to be impossible in the tests during the t=5-15 minute period when satellite communication was very poor. DTN's store and forward capability allowed COC to continue to update POP even when satcomm throughput was very intermittent, and the higher-level server systems did not need to go through contact reinitialization.

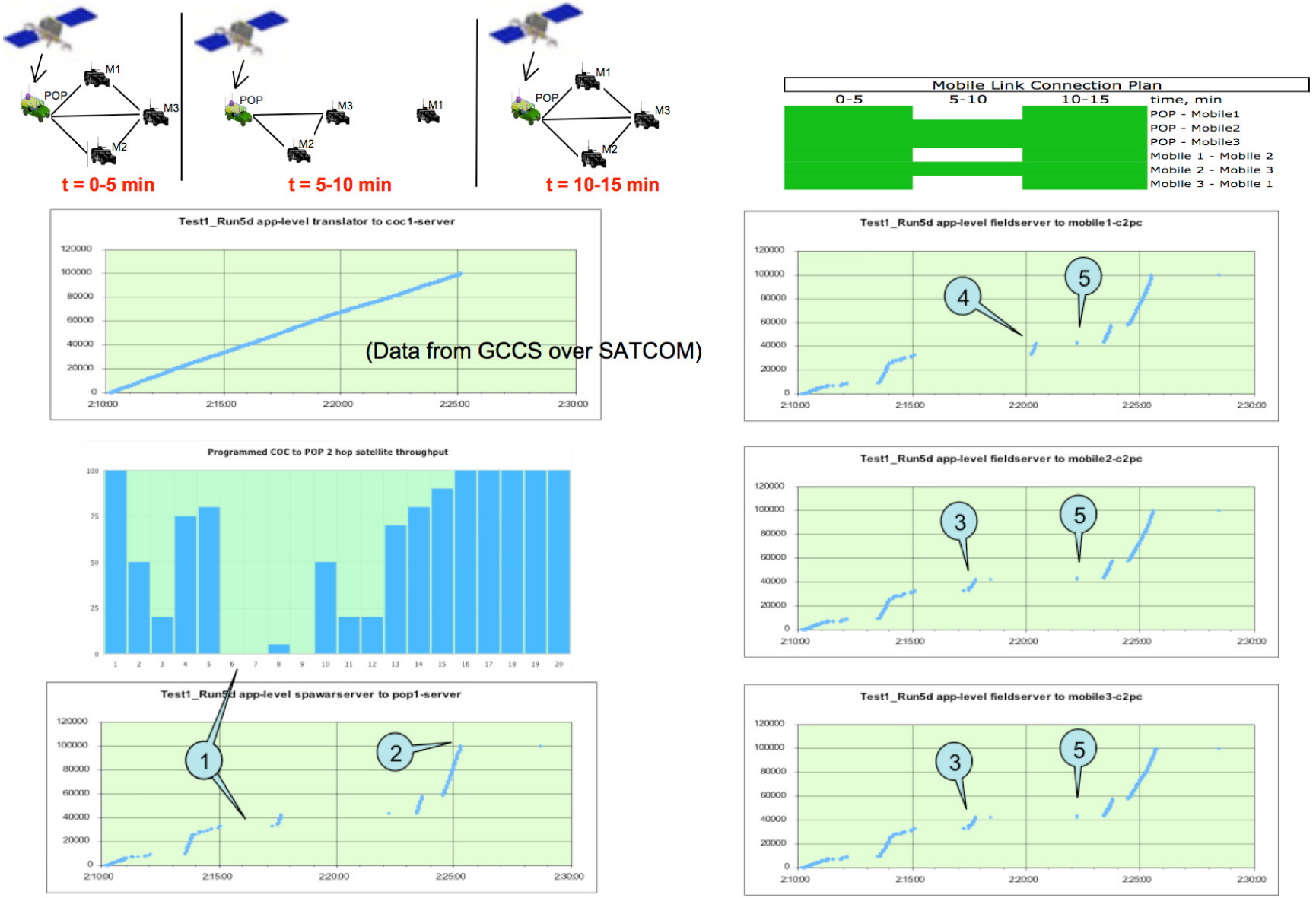


Figure 21: MSCTSSA Scenario 1 Run.

Figure 22 shows the SIE/SN/DTN performance during a MCTSSA Scenario 6 run. In this case, the three mobile units move from a mesh network (where all can see POP), to a linear arrangement where POP can only see mobile1, and the mobile units can only see the vehicle in front or behind. During the movement, connections between vehicles are broken then regained as the scenario progresses (see timeline upper right).

In this scenario, the blue call-out notes are as follows:

1. Satellite outages cause gaps in POP data available to send to mobile1-mobile3
2. mobile3 getting data forwarded from mobile2 until it moves out of range
3. mobile2 getting data from mobile 1 until out of range
4. mobile1 looses contact with POP
5. mobile3 and mobile2 back in range of one another; mobile3 gets updates from mobile 2
6. mobile1 and mobile2 in range, mobile2 gets updates from mobile1
7. mobile3 gets updates from mobile1 through mobile2
8. Finally everyone in linear array, DTN gets data to everyone through its store & forward capability.

In the C2PC case, mobile1 received updates while it was in reliable contact with POP, but mobile2 and mobile3 stopped getting data as soon as they were out of range of POP, since there is no multicast forwarding mechanism in C2PC.

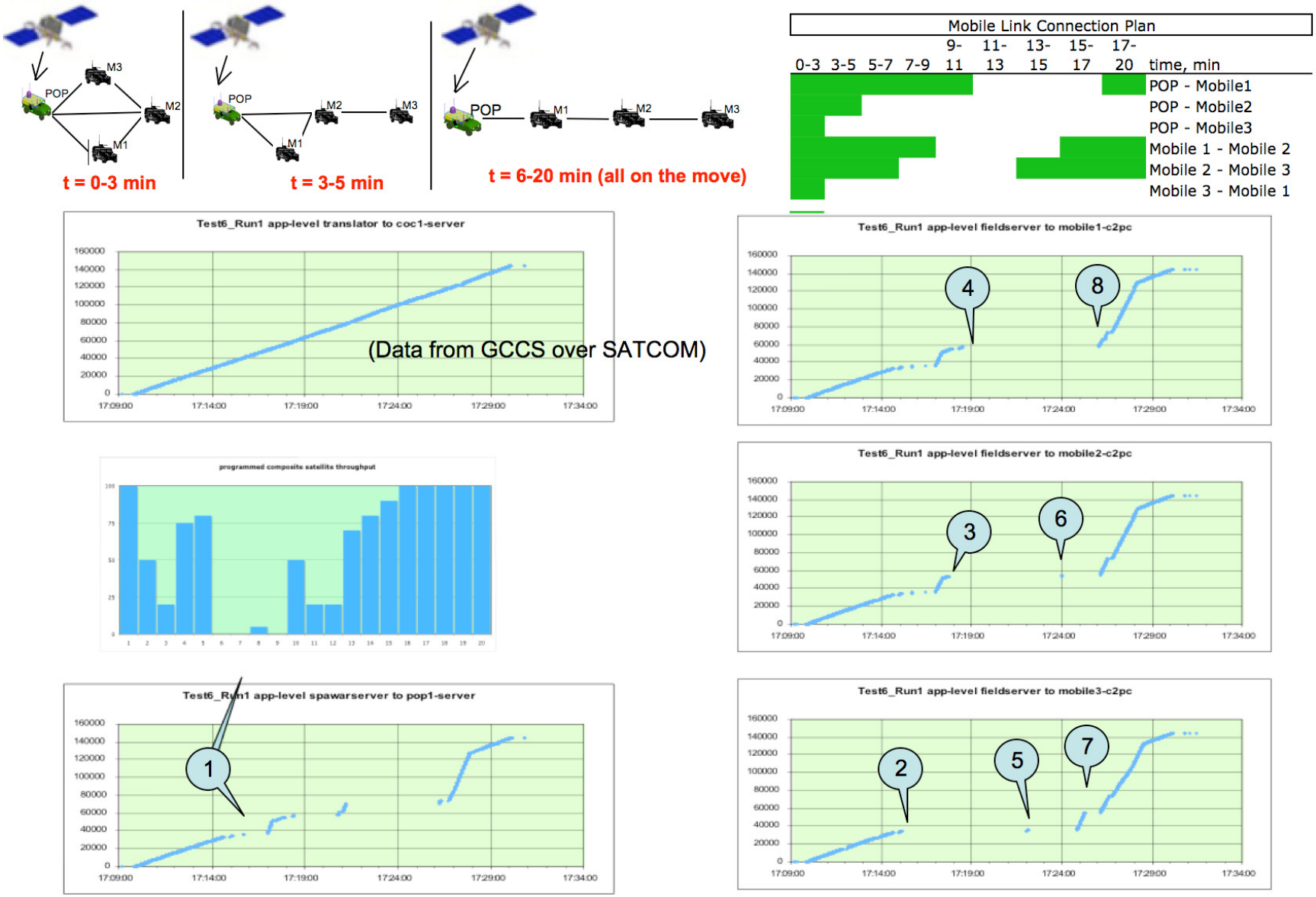


Figure 22: MCTSSA Scenario 6 Run

3.2.3.4.5 Test Anomalies and Analysis Difficulties

It was discovered that the GCCS-SIE translator was very inefficient, and was causing the SIE/SN/DTN traffic over the satellite links to be six times greater than expected. Investigations and discussions with the developers at SPAWAR revealed that a GCCS track update containing 6 track parameters would result in one C2PC message, but it the SPAWAR GCCS-SIE translator would create six SIE/SN messages to do the same update over SharedNet. This made the link utilization statistics of SIE/SN/DTN look poor as compared to C2PC. The problem is understood, and it our understanding that SPAWAR is working on a revised and more efficient version of the GCCS-SIE translator.

In retrospect, the comparison of C2PC to SIE/SN was somewhat unbalanced. C2PC is an unreliable system (that is to say reliable delivery to clients is not assured), whereas SIE/SN is a reliable delivery system, and enforces in-order, gapless delivery of track data to the user-level display mechanisms.. The difference in the reliability philosophies between the two systems made C2PC appear to be working better under some circumstances, but what was being observed was later C2PC tracks making it through (with lots of data loss), while the SIE/SN/DTN system buffered data up while awaiting retransmission of lost messages in order to assure complete and in-order data flow to the operator's visual display.

Lack of quantifiable C2PC performance data, and inability to correlate the track identifications of data being sent via SIE/SN and C2PC made it impossible to assess performance metrics such as track update latency, or even number of tracks received by C2PC. Raw data dumps are being evaluated by the other teams, and should there be additional funding to continue the quantitative analysis, JPL SN and PTL developers have a software design that would allow for the needed correlation information to be dissected from the Wireshark logs.

(A comparative analysis between C2PC and SN (Classic or DTN versions) has been very difficult, as attested to by other analysts in the DARPA team. Some of the reasons are cited below, quoted from email traffic with the team:

> *1. C2PC developers state that C2PC logs are designed only for internal consistency checks; they are not suitable instrumentation for performance checking.*
>
> *2. Hence the only instrumentation data available for C2PC performance quantification is Wireshark data, which is by definition **on the wire**. Put differently, there is no quantitative C2PC data that can be automatically collected at the application/user layer.*
>
> *3. In contrast, SN (both flavors) has extensive instrumentation logging **at the top of the network stack and at the application/user layer**.*

*4. However, during the transition tests, no Wireshark dissector existed for SN (either flavor), and <bold>**there is no simple way to relate the data in SN packets on the wire or at the top of the network stack to SN objects at the application/user layer**.*

5. Hence, without a SN dissector, no apples-to-apples comparison can be made between C2PC data and SN data.

By the time the need for better instrumentation for C2PC and Wireshark dissector tools for SharedNet was identified, there was neither sufficient time nor DARPA budget to do further software development at JPL.)

3.2.3.5 Overall Test Summary

Comparison between the systems tested in both Hawaii and in the PTL Lab for MCTSSA encompasses a number of metrics such as bandwidth utilization, data latency, throughput, total amount of data reaching the intended destinations, routing and IP neighbor discover performance, and performance of the NORM protocol.

The details of these metrics are being reported by the SPAWAR/BBN/MITRE teams. From the JPL observer standpoint, the main qualitative result that was clearly evident throughout the tests is that DTN clearly outperforms standard IP-based delivery systems in a disrupted environment. The legacy systems compared to SIE/SN/DTN failed to handle satellite disruptions or breaks in mobile connectivity to the MIP or POP server, while the DTN-enabled system got the data through.

The tests showed that the NRL Simple Multicast Forwarding system (explored during the MCBH field tests) underperformed expectations, and qualitatively did not compare well to DTN in the ability to store and forward data in a linear topology. Further baseline tests of SMF vs. DTN would have to be conducted to accurately quantify the relative performances of the two protocols in a linear topology.

An enormous amount of data was collected, and much analysis work remains. This analysis work will be crucial to enable the optimization of the performance of the system, inasmuch as there are reliability and timing elements at every level in the protocol stack fielded; while we obtained generally acceptable results, there may be a lot of improvement that can be made with suitable parameter adjustments in the reliability and retransmission mechanisms.

The ease with which DTN handled the satellite disruptions was noteworthy, and the MCTSSA representatives expressed a desire to field a general-purpose DTN proxy node at either end of a satellite link (possibly including a TCP/IP performance-enhancing proxy, or PEP). We did some informal demonstrations of DTN-enabled web browsing and chat while the MCTSSA staff was in the PTL, showing that there are many applications besides SIE/SN or C2PC that would benefit from DTN.

All-in-all, the DTN protocol was successfully demonstrated and tested over operational networks in the field, and the benefits and advantages of DTN were very easy to see.

3.2.4 USMC Tactical Systems Support Activity Report and Conclusions

MCTSSA Tactical Networks Chief Engineer Captain John Frushour, actively participated in all testing at JPL, and his observations and conclusions were reported in Reference [3]. The key quotation from his Executive Summary is *"Overall, the contrast between DTN and non-DTN enhanced network testing showed overwhelmingly positive results. Where non-DTN networks could not deliver traffic to disconnected nodes, the DTN-enhanced networks were nearly 100% reliable."*

During the MCTSSA testing, JPL made lab space available for Capt. Frushour and others to do conduct testing of a commercial Performance-Enhancing Proxy known as the Web Assured Response Protocol (WARP) and to assess the use of DTN with this product. The results of these tests, as reported in Reference [4], are best summed up in that report's Executive Summary, *"Independently, each product is successful in their own way. During this interoperability test, the goal was to examine each technologies approach to "cleaning up" satellite communications and determine whether the two technologies exhibit a complementary relationship. The results were quite attractive. Not only were the products able to be seamlessly integrated in a short time; the benefits were instantaneous and tangible."*

Finally, Captain Frushour recommended future development and test activities (Ref. [3]) that warrant citation in this report:

> *Suggestions for Development*
>
> *a. Client Application (PC based)*
> *Previously (from Hawaii trip report) it was stated that there are two suggested means of development for the DTN architecture. This can be refined. Given that DTN has an associated API and the relative ease of integration seen in the WARP/DTN 7*

integration testing also performed at NASA JPL, it would be wise to advance DTN's development in this fashion:
1) Decouple any application dependency between DTN and products such as SharedNet.

2) Investigate the possibility of integrating the Bundle Protocol Agent, or a similar DTN implementation, with WAN acceleration technologies. This would henceforth provide both acceleration AND reliability to troublesome WAN links over satellite or terrestrial systems.

3) Investigate the feasibility of application specific proxy devices for the DTN architecture. Further traffic analysis would have to be garnered in order to determine which proxy devices warrant incorporation into the DTN architecture.

4) Develop an integrated product, preferably a client application (although appliance is certainly not prohibited), that is PC based and can be configured via traditional Microsoft Active Directory OU policies.

b. Network Topology

Future testing of the DTN architecture needs to be of the integrated variety. Because DTN is not a traditional performance enhancing proxy, and also its approach is both novel and unique, it should be tested in order to determine the maximum benefit over existing systems. Additionally, it would be wise to develop realistic models of current traffic flows, so as to compare them with DTN performance benefits and ultimately compose a more realistic traffic analysis.

3.2.5 Lessons Learned

Comments and lessons learned thoughts below are meant to be reminders and observations to consider for the next similar project, and are in no way meant as criticism of any group or program.

1. Unexpected Data Volume

For each test run, the data collected from each node in the test included log data from SN, BPA, MGEN, NORM and a Wireshark capture of all traffic. Test data collection, instrumentation and post-test analysis steps were simply to instrument everything, take pcap dumps of traffic from all 20 nodes and all ports on each node and hope to sort out the data later.

The SN log analysis runs alone produced 3,178 files and many hundreds of plots.

In future test programs such as this, it is recommended that more time be spent planning on what metrics are to be determined, what data are to be collected, what analysis software is necessary (and what has to be written) and how much analysis time and effort will be needed to sort through the data.

2. Integration Time and Effort

   Several items in this category are worth noting:

- Insufficient time was allocated to get the basic systems up and running, and this took away from time intended to do dry runs and parameter tuning of the SN, BPA, and NORM components.
- The difficulty in configuration and complexity of running Win and Linux at the same time was underestimated.
- If government-furnished software is to be used, make sure it is working properly before the installation team departs – it took several days of lost time trying to make C2PC and GCCS work properly, and it was found that the C2PC installation wasn't correct.
- Multi-team integration of software needs face-to-face participation in the lab initially. Once the basic components can be demonstrated to work, the team can disperse to their home labs and do the tuning and testing of the system remotely.
- Robust remote access is necessary, and a central data / SVN / Wiki repository like BBN provided is crucial. Some company firewall problems required workarounds to allow developers to log in remotely to run tests or change configurations. Time spent planning these details is time well spent.

3. Use of Government (or Commercial) "Off-the-shelf" Software

- Government or Commercial Off the Shelf products can't be instrumented easily (or at all). For example, C2PC has no log files, no performance indications (such as number of tracks received), and the on-the-wire formats of C2PC data had to be "reverse engineered" using Wireshark data exported as PDML files.
- GOTS/COTS may not be able to accommodate controlled tests – e.g. the GCCS generator software couldn't be programmed to stop after a certain number of track updates or after a certain time.
- GOTS software operations need to be considered carefully in test planning. For example, when a C2PC client goes out of comm , it must be manually reconnected to the server; this was not always possible to do in a timely (or predictable) fashion during the MCTSSA tests.

3.3 Contact information

For further information please contact the JPL Core Engineering Support Team:

Scott Burleigh – Scott.Burleigh@jpl.nasa.gov
Rick Borgen – Richard.L.Borgen@jpl.nasa.gov
James McKelvey James.W.Mckelvey@jpl.nasa.gov
John Segui - John.S.Segui@jpl.nasa.gov
Leigh Torgerson – ltorgerson@jpl.nasa.gov
Philip Tsao - Philip.C.Tsao@jpl.nasa.gov

3.4 References

[1] Software Interoperability Environment (SIE) And Disruption Tolerant Networking (DTN) Test Plan, 13 January 2010, Version 1.1 – David Durham, SPAWAR

[2] Software Interoperability Environment (SIE) And Disruption Tolerant Networking (DTN) MCTSSA Test Plan, 09 February 2010, Version 0.9 – David Durham, SPAWAR

[3] TRIP REPORT: DISRUPTION TOLERANT NETWORK (DTN) TEST, NASA JET PROPULSION FACILITY (JPL) (U), 24 February 2010, Capt. J. Frushour, USMC, MCTSSA

[4] WARP/DTN - Web Assured Response Protocol / Disruption Tolerant Networking Protocol Quick Look (FOUO), 23 February 2010, MCTSSA Program and Engineering Support Group

The research described in this publication was carried out by the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by DARPA.

Appendix A: Acronyms and Abbreviations

ATH	At-the-Halt
C2PC	Command and Control Personal Computer
C4I	Command, Control, Communications, Computers, Intelligence
DARPA	Defense Advanced Research Projects Agency
DARS	Data Archive and Replay System
DTN	Disruption Tolerant Networking
GCCS	Global Command and Control System
IOS	Information-Operations Server
M2C2	Mobile Modular Command and Control
MARFORPAC	Marine Forces Pacific
MCTSSA	Marine Corps Tactical Systems Support Activity
MEC	Marine Experimentation Center
MCBH	Marine Corps Base Hawaii
MIP	Mobile Modular Command and Control (M2C2) Interoperability Prototype (MIP)
NAI	Named Area of Interest
OTH	On-The-Horizon
OTM	On-The-Move
ONR	Office of Naval Research
PEP	Performance Enhancing Proxy
POP	Point of Presence
POR	Program of Record
SIE	Software Interoperability Environment
SA	Situation Awareness
SN	SharedNet
SVN	Subversion version control system

Appendix B: MCTSSA - WANem setup

(by Phil Tsao)

WANem satellite channel simulation set up instructions:

Background:
WANem is a "liveCD" Linux distribution that turns almost x86 computer into a basic link emulator. (adjustable packet loss, throughput, delay, etc)

Often times, it may be desirable to emulate more than one link on a single computer. Assuming the physical machine has enough ram and ethernet ports, one may simply run multiple WANem instances in multiple virtual machines. Multiple virtual machines may represent a undesirable overhead and are not necessary when one wishes to implement multiple independent ethernet bridges on disjoint subnets.

Example:
COC (10.1.100.166 netmask 255.255.255.0) <-> WANem <-> TEP (214.15.3.20 netmask 255.255.255.248) <-> WANem <-> POP (214.15.3.85 netmask 255.255.255.240)

This configuration requires two virtual machines each of which implements a router.

1. Populate a computer with a hard drive and 5 ethernet NICs. I assume the 5 NICs correspond to eth0 through eth4. If they don't on your machine, adjust the the instructions as needed. I further assume eth1 goes to COC, eth2 goes to the left side of TEP, eth3 goes to the right side of TEP and eth4 goes to POP.
2. Install a relatively recent Linux distribution (Ubuntu Jaunty Jackalope used for MCTSSA testing)
3. Assume eth0 is attached to the administrative LAN. Assign an ip (ifconfig eth0 192.168.3.101 netmask 255.255.255.0) and edit the appropriate config files to make it persist across reboots.
4. Install VirtualBox. (Other VM software will likely work)
5. Depending on your package manager, you may need to reboot. There is an "internal" administrative LAN which will be used to control the VM's. Assign an ip (ifconfig vbox0 192.168.4.1 netmask 255.255.255.0) and edit the appropriate config files to make it stick.
6. In VirtualBox, create 2 identical VM's. Have them both boot from a WANem liveCD (preferably an iso image). Enable 3 network adapters: Assign eth0 of the guest to the host only adapter and eth1 and eth2 to bridged adapters (eth1, eth2, eth3 and eth4 respectively).
7. Start up the first VM and wait until it gives you the WANem prompt. Type 'exit2shell' to get to the bash prompt.
8. Assign ip addresses (ifconfig eth0 192.168.4.2 netmask 255.255.255.0 && ifconfig eth1 10.1.100.165 netmask 255.255.255.0 && ifconfig eth2 214.15.3.17 netmask 255.255.255.248)
9. Make sure COC and TEP can ping across the bridge (assuming COC and TEP have routes set appropriately. You may want to kill the "pump" process to save future aggravation in case DHCP servers are running.
10. If necessary, restart apache in /etc/init.d/apache2

11. On the host, point your web browser to <http://192.168.4.2/WANem>. If you get an error message, you must have missed a step or mistyped something.
12. Now would be a good time to "save snapshot" under the "machine" menu.
13. If you don't have "socat" on the host, install it. Put something like "socat tcp-listen:8001,fork tcp:192.168.4.2:80 & socat tcp-listen:2001,fork tcp:192.168.4.2:22 &" in /etc/rc.local of the host unless you like typing in long cryptic command lines every time the computer reboots.
14. If you did everything correctly, you should be able to access the VM by pointing a web browser to <http://192.168.3.101:8001/WANem> or by SSH (ssh -vc arcfour -p 2001 root@192.168.3.101)
15. Repeat steps 7-13 for the second VM. The ip addresses are different (ifconfig eth0 192.168.4.3 netmask 255.255.255.0 && ifconfig eth1 214.15.3.18 netmask 255.255.255.248 && ifconfig 214.15.3.81 netmask 255.255.255.240) and the socat commands are different (socat tcp-listen:8001,fork tcp:192.168.4.2:80 & socat tcp-listen:2001,fork tcp:192.168.4.2:22 &)

If you need to reboot the host, be sure to revert guests to the previous saved state before starting them up to avoid having to perform the above steps again.

Example:
COC (10.1.100.166 netmask 255.255.255.0) <-> WANem <-> TEP (214.15.3.20 netmask 255.255.255.248)
COC2 (10.1.100.29 netmask 255.255.255.0) <-> WANem <-> TEP2 (10.1.100.30 netmask 255.255.255.0)

In this case steps 1 through 14 are pretty much same as before. The only difference is the second VM is a bridge and not a router. For the second VM replace step 8 with...

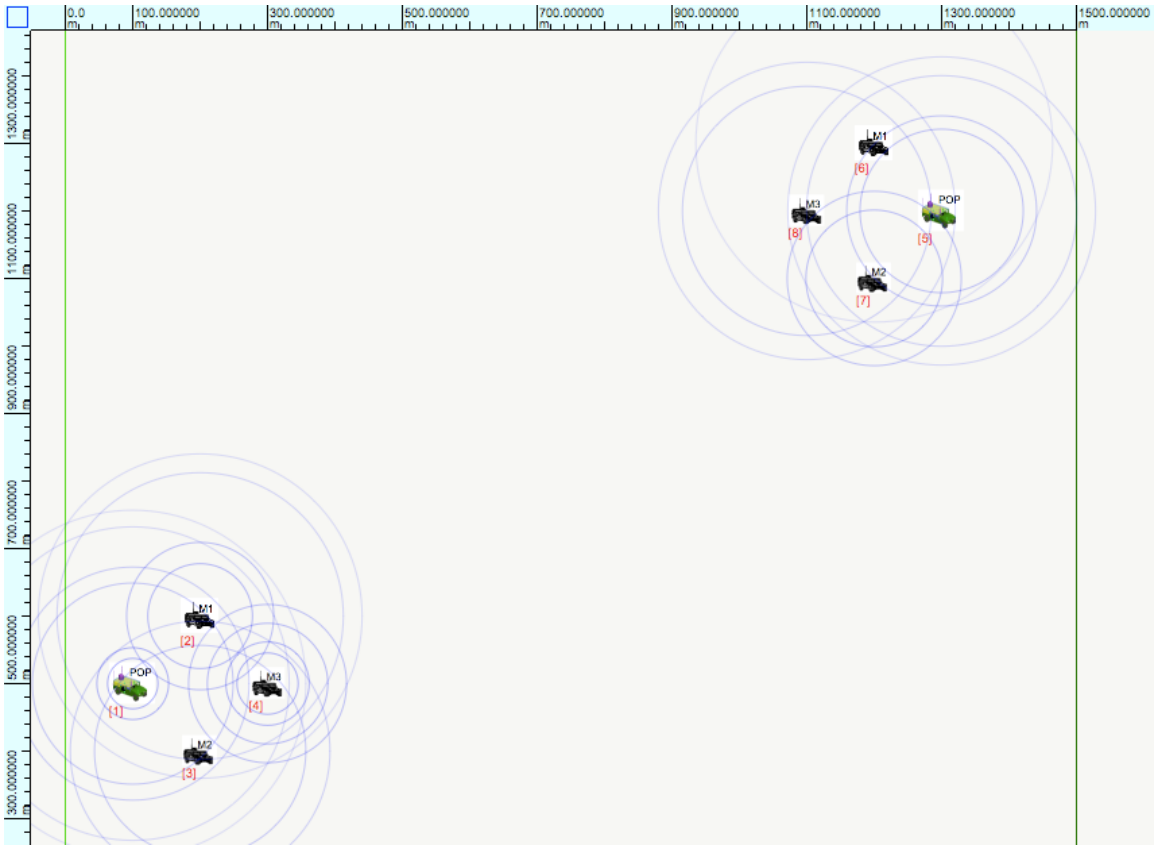
8. Create a bridge (brctl addbr br0) and add interfaces to it (brctl addif br0 eth1 && brctl addif br0 eth2). Enable STP (brctl stp br0 on) and turn on the bridge (ifconfig br0 up). You should be able to ping across in a few moments.

Appendix C – MCTSSA Test Mobile Network Simulation/Emulation

(by John Segui)

3.1 Scenario Configuration and Models

a. Topology



Two identical 802.11b wireless subnets were created; each with one Point of Presence (POP) node and three wireless clients. One subnet reserved for SharedNet Classic/DTN and another for C2PC Native. Terrain was modeled as a flat 1500m X 1500m grid. All nodes were at ground level. Two-way path-loss was modeled

b. Wireless

All nodes used 802.11b Physical and Link layers in DCF (ad-hoc) mode fixed at 11Mbps rate allowing any-to-any communication without a base station. All radios used omni-directional antennas at a height of 1.5 meters and 15dbm transmit power.

The channel, radio and MAC settings are provided in section 1.3.

c. Mobility

Nodes used a 1 meter position granularity and moved according to the position schedules provided in Appendix B. Nodes would drive at speeds necessary to get from point A to point B in (timeStampB – timeStampA) seconds.

Full node position schedules are provided in section 1.4.

d. Network Settings

Static routes were used. No routing protocol was modeled. IPv4 was modeled with the settings in section 1.5.

3.2 Emulator/Simulator Machine Software Configuration

a. Operating System

Debian Etch. Uname output:

Linux debian 2.6.18-6-686 #1 SMP Thu Nov 5 16:28:13 UTC 2009 i686 GNU/Linux

b. Java

Java version "1.5.0_14"

Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_14-b03)

Java HotSpot(TM) Server VM (build 1.5.0_14-b03, mixed mode)

c. QualNet

4.5.1 with the developer, multimedia enterprise, wireless, advanced wireless, urban propagation, and IP Network Emulator (IPNE) libraries installed.

The ipnetworkemulator.cpp source file was extended to allow for simultaneous MULTICAST and NAT-NO options in the IPNE.

d. Network Configuration

To enable routing on the network emulator box and handle multiple interfaces with identical subnet addresses, static routes were added. Different routes were needed for SharedNet classic (windows) and SharedNet DTN (Linux) shown in Appendix D.

e. Machine Configuration

Each machine participating in the simulated wireless network was directly crossover connected to the emulator/simulator machine. Consequently, each machine had to be configured to direct all traffic destined for other participants through the emulator/simulator. The Windows and Linux routing tables were configured for this. Additionally, the Linux virtual machines were configured with iptables rules to translate 234.1.1.1 multicast traffic to appropriate 225.x.y.z addresses. The 22.5.x.y.z addresses were needed to differentiate traffic from/to each node/interface on the emulator machine.

The configuration scripts are provided in section 1.6.

3.3 Radio, Channel and MAC Model Settings

```
#Irrelevant configuration lines omitted for brevity
# ************ Channel **********
PROPAGATION-CHANNEL-FREQUENCY 2400000000
PROPAGATION-MODEL STATISTICAL
PROPAGATION-LIMIT -111.0
PROPAGATION-PATHLOSS-MODEL TWO-RAY
PROPAGATION-SHADOWING-MODEL CONSTANT
PROPAGATION-SHADOWING-MEAN 4.0
PROPAGATION-FADING-MODEL NONE
# ************ Radio/Physical Layer **********
PHY-MODEL PHY802.11b
PHY802.11-AUTO-RATE-FALLBACK NO
PHY802.11-DATA-RATE 11000000
PHY802.11b-TX-POWER-11MBPS 15.0
PHY802.11b-RX-SENSITIVITY-11MBPS -83.0
PHY-RX-MODEL PHY802.11b
PHY-TEMPERATURE 290.0
PHY-NOISE-FACTOR 10.0
ANTENNA-MODEL OMNIDIRECTIONAL
ANTENNA-GAIN 0.0
ANTENNA-HEIGHT 1.5
ANTENNA-EFFICIENCY 0.8
ANTENNA-MISMATCH-LOSS 0.3
ANTENNA-CABLE-LOSS 0.0
ANTENNA-CONNECTION-LOSS 0.2
# ************ MAC Protocol **********
MAC-PROTOCOL MACDOT11
MAC-DOT11-DIRECTIONAL-ANTENNA-MODE NO
MAC-DOT11-SHORT-PACKET-TRANSMIT-LIMIT 7
MAC-DOT11-LONG-PACKET-TRANSMIT-LIMIT 4
MAC-DOT11-RTS-THRESHOLD 0
MAC-DOT11-ASSOCIATION NONE
MAC-DOT11-IBSS-SUPPORT-PS-MODE NO
MAC-PROPAGATION-DELAY 1US
PROMISCUOUS-MODE YES
```

3.4 – Node Mobility

Format: **nodeNumber timeOfMove** (**X**, **Y**, **Z**) 0 0

3.4.1 Scenario 0

1 0 (100.0, 500.0, 0.0) 0 0
2 0 (200.0, 600.0, 0.0) 0 0
3 0 (200.0, 400.0, 0.0) 0 0
4 0 (300.0, 500.0, 0.0) 0 0

5 0 (1300.0, 1200.0, 0.0) 0 0
6 0 (1200.0, 1300.0, 0.0) 0 0
7 0 (1200.0, 1100.0, 0.0) 0 0
8 0 (1100.0, 1200.0, 0.0) 0 0

3.4.2 Scenario 1

1 0 (100.0, 500.0, 0.0) 0 0
2 0 (200.0, 600.0, 0.0) 0 0
3 0 (200.0, 400.0, 0.0) 0 0
4 0 (300.0, 500.0, 0.0) 0 0
5 0 (1300.0, 1200.0, 0.0) 0 0
6 0 (1200.0, 1300.0, 0.0) 0 0
7 0 (1200.0, 1100.0, 0.0) 0 0
8 0 (1100.0, 1200.0, 0.0) 0 0
2 10M (200.0, 600.0, 0.0) 0 0
2 11M (600.0, 600.0, 0.0) 0 0
2 15M (600.0, 600.0, 0.0) 0 0
2 16M (200.0, 600.0, 0.0) 0 0
6 10M (1200.0, 1300.0, 0.0) 0 0
6 11M (800.0, 1300.0, 0.0) 0 0
6 15M (800.0, 1300.0, 0.0) 0 0
6 16M (1200.0, 1300.0, 0.0) 0 0

3.4.3 Scenario 2

1 0 (100.0, 500.0, 0.0) 0 0
2 0 (200.0, 600.0, 0.0) 0 0
3 0 (200.0, 400.0, 0.0) 0 0
4 0 (300.0, 500.0, 0.0) 0 0
5 0 (1300.0, 1200.0, 0.0) 0 0
6 0 (1200.0, 1300.0, 0.0) 0 0
7 0 (1200.0, 1100.0, 0.0) 0 0
8 0 (1100.0, 1200.0, 0.0) 0 0
2 10M (200.0, 600.0, 0.0) 0 0
2 11M (600.0, 600.0, 0.0) 0 0
2 15M (600.0, 600.0, 0.0) 0 0
2 16M (400.0, 600.0, 0.0) 0 0
6 10M (1200.0, 1300.0, 0.0) 0 0
6 11M (800.0, 1300.0, 0.0) 0 0
6 15M (800.0, 1300.0, 0.0) 0 0

6 16M (1000.0, 1300.0, 0.0) 0 0

3.4.4 Scenario 3

1 0 (100.0, 500.0, 0.0) 0 0
2 0 (200.0, 600.0, 0.0) 0 0
3 0 (200.0, 400.0, 0.0) 0 0
4 0 (300.0, 500.0, 0.0) 0 0
5 0 (1300.0, 1200.0, 0.0) 0 0
6 0 (1200.0, 1300.0, 0.0) 0 0
7 0 (1200.0, 1100.0, 0.0) 0 0
8 0 (1100.0, 1200.0, 0.0) 0 0
2 10M (200.0, 600.0, 0.0) 0 0
2 11M (500.0, 500.0, 0.0) 0 0
2 15M (500.0, 500.0, 0.0) 0 0
2 16M (700.0, 500.0, 0.0) 0 0
3 15M (200.0, 400.0, 0.0) 0 0
3 16M (300.0, 500.0, 0.0) 0 0
4 15M (300.0, 500.0, 0.0) 0 0
4 16M (500.0, 500.0, 0.0) 0 0
6 10M (1200.0, 1300.0, 0.0) 0 0
6 11M (900.0, 1200.0, 0.0) 0 0
6 15M (900.0, 1200.0, 0.0) 0 0
6 16M (700.0, 1200.0, 0.0) 0 0
7 15M (1200.0, 1100.0, 0.0) 0 0
7 16M (1100.0, 1200.0, 0.0) 0 0
8 15M (1100.0, 1200.0, 0.0) 0 0
8 16M (900.0, 1200.0, 0.0) 0 0

3.4.5 Scenario 4

1 0 (100.0, 500.0, 0.0) 0 0
2 0 (300.0, 500.0, 0.0) 0 0
3 0 (500.0, 500.0, 0.0) 0 0
4 0 (700.0, 500.0, 0.0) 0 0
5 0 (1400.0, 1200.0, 0.0) 0 0
6 0 (1200.0, 1200.0, 0.0) 0 0
7 0 (1000.0, 1200.0, 0.0) 0 0
8 0 (800.0, 1200.0, 0.0) 0 0
4 7M (700.0, 500.0, 0.0) 0 0
4 7.5M (900.0, 500.0, 0.0) 0 0
4 9M (900.0, 500.0, 0.0) 0 0
4 9.5M (1100.0, 500.0, 0.0) 0 0
4 11M (1100.0, 500.0, 0.0) 0 0
4 11.5M (1300.0, 500.0, 0.0) 0 0
3 9M (500.0, 500.0, 0.0) 0 0
3 9.5M (700.0, 500.0, 0.0) 0 0

3 11M (700.0, 500.0, 0.0) 0 0
3 11.5M (900.0, 500.0, 0.0) 0 0
3 13M (900.0, 500.0, 0.0) 0 0
3 13.5M (1100.0, 500.0, 0.0) 0 0
2 11M (300.0, 500.0, 0.0) 0 0
2 11.5M (500.0, 500.0, 0.0) 0 0
2 13M (500.0, 500.0, 0.0) 0 0
2 13.5M (600.0, 500.0, 0.0) 0 0
2 15M (600.0, 500.0, 0.0) 0 0
2 15.5M (900.0, 500.0, 0.0) 0 0
1 13M (100.0, 500.0, 0.0) 0 0
1 13.5M (200.0, 500.0, 0.0) 0 0
1 15M (200.0, 500.0, 0.0) 0 0
1 15.5M (300.0, 500.0, 0.0) 0 0
1 17M (300.0, 500.0, 0.0) 0 0
1 17.5M (700.0, 500.0, 0.0) 0 0
8 7M (800.0, 1200.0, 0.0) 0 0
8 7.5M (600.0, 1200.0, 0.0) 0 0
8 9M (600.0, 1200.0, 0.0) 0 0
8 9.5M (400.0, 1200.0, 0.0) 0 0
8 11M (400.0, 1200.0, 0.0) 0 0
8 11.5M (200.0, 1200.0, 0.0) 0 0
7 9M (1000.0, 1200.0, 0.0) 0 0
7 9.5M (800.0, 1200.0, 0.0) 0 0
7 11M (800.0, 1200.0, 0.0) 0 0
7 11.5M (600.0, 1200.0, 0.0) 0 0
7 13M (600.0, 1200.0, 0.0) 0 0
7 13.5M (400.0, 1200.0, 0.0) 0 0
6 11M (1200.0, 1200.0, 0.0) 0 0
6 11.5M (1000.0, 1200.0, 0.0) 0 0
6 13M (1000.0, 1200.0, 0.0) 0 0
6 13.5M (900.0, 1200.0, 0.0) 0 0
6 15M (900.0, 1200.0, 0.0) 0 0
6 15.5M (600.0, 1200.0, 0.0) 0 0
5 13M (1400.0, 1200.0, 0.0) 0 0
5 13.5M (1300.0, 1200.0, 0.0) 0 0
5 15M (1300.0, 1200.0, 0.0) 0 0
5 15.5M (1200.0, 1200.0, 0.0) 0 0
5 17M (1200.0, 1200.0, 0.0) 0 0
5 17.5M (800.0, 1200.0, 0.0) 0 0

3.4.6 Scenario 5

1 0 (100.0, 500.0, 0.0) 0 0
2 0 (300.0, 450.0, 0.0) 0 0
3 0 (500.0, 500.0, 0.0) 0 0

4 0 (700.0, 500.0, 0.0) 0 0
5 0 (1400.0, 1200.0, 0.0) 0 0
6 0 (1200.0, 1150.0, 0.0) 0 0
7 0 (1000.0, 1200.0, 0.0) 0 0
8 0 (800.0, 1200.0, 0.0) 0 0
4 7M (700.0, 500.0, 0.0) 0 0
4 7.5M (1300.0, 500.0, 0.0) 0 0
4 15M (1300.0, 500.0, 0.0) 0 0
4 15.5M (1100.0, 500.0, 0.0) 0 0
4 17M (1100.0, 500.0, 0.0) 0 0
4 17.5M (300.0, 550.0, 0.0) 0 0
3 9M (500.0, 500.0, 0.0) 0 0
3 9.5M (1100.0, 500.0, 0.0) 0 0
3 11M (1100.0, 500.0, 0.0) 0 0
3 11.5M (500.0, 500.0, 0.0) 0 0
3 13M (500.0, 500.0, 0.0) 0 0
3 13.5M (1100.0, 500.0, 0.0) 0 0
3 15M (1100.0, 500.0, 0.0) 0 0
3 15.5M (500.0, 500.0, 0.0) 0 0
8 7M (800.0, 1200.0, 0.0) 0 0
8 7.5M (200.0, 1200.0, 0.0) 0 0
8 15M (200.0, 1200.0, 0.0) 0 0
8 15.5M (400.0, 1200.0, 0.0) 0 0
8 17M (400.0, 1200.0, 0.0) 0 0
8 17.5M (1200.0, 1250.0, 0.0) 0 0
7 9M (1000.0, 1200.0, 0.0) 0 0
7 9.5M (400.0, 1200.0, 0.0) 0 0
7 11M (400.0, 1200.0, 0.0) 0 0
7 11.5M (1000.0, 1200.0, 0.0) 0 0
7 13M (1000.0, 1200.0, 0.0) 0 0
7 13.5M (500.0, 1200.0, 0.0) 0 0
7 15M (500.0, 1200.0, 0.0) 0 0
7 15.5M (1000.0, 1200.0, 0.0) 0 0

3.4.7 Scenario 6 (combination of scenarios 3 and 4)

1 0 (100.0, 500.0, 0.0) 0 0
2 0 (200.0, 400.0, 0.0) 0 0
3 0 (300.0, 500.0, 0.0) 0 0
4 0 (200.0, 600.0, 0.0) 0 0
5 0 (1400.0, 1200.0, 0.0) 0 0
6 0 (1300.0, 1100.0, 0.0) 0 0
7 0 (1200.0, 1200.0, 0.0) 0 0
8 0 (1300.0, 1300.0, 0.0) 0 0
2 5M (200.0, 400.0, 0.0) 0 0
2 5.5M (300.0, 500.0, 0.0) 0 0

3 5M (300.0, 500.0, 0.0) 0 0
3 5.5M (500.0, 500.0, 0.0) 0 0
4 3M (200.0, 600.0, 0.0) 0 0
4 3.5M (500.0, 500.0, 0.0) 0 0
4 5M (500.0, 500.0, 0.0) 0 0
4 5.5M (700.0, 500.0, 0.0) 0 0
4 7M (700.0, 500.0, 0.0) 0 0
4 7.5M (900.0, 500.0, 0.0) 0 0
4 9M (900.0, 500.0, 0.0) 0 0
4 9.5M (1100.0, 500.0, 0.0) 0 0
4 11M (1100.0, 500.0, 0.0) 0 0
4 11.5M (1300.0, 500.0, 0.0) 0 0
3 9M (500.0, 500.0, 0.0) 0 0
3 9.5M (700.0, 500.0, 0.0) 0 0
3 11M (700.0, 500.0, 0.0) 0 0
3 11.5M (900.0, 500.0, 0.0) 0 0
3 13M (900.0, 500.0, 0.0) 0 0
3 13.5M (1100.0, 500.0, 0.0) 0 0
2 11M (300.0, 500.0, 0.0) 0 0
2 11.5M (500.0, 500.0, 0.0) 0 0
2 13M (500.0, 500.0, 0.0) 0 0
2 13.5M (600.0, 500.0, 0.0) 0 0
2 15M (600.0, 500.0, 0.0) 0 0
2 15.5M (900.0, 500.0, 0.0) 0 0
1 13M (100.0, 500.0, 0.0) 0 0
1 13.5M (200.0, 500.0, 0.0) 0 0
1 15M (200.0, 500.0, 0.0) 0 0
1 15.5M (300.0, 500.0, 0.0) 0 0
1 17M (300.0, 500.0, 0.0) 0 0
1 17.5M (700.0, 500.0, 0.0) 0 0
6 5M (1300.0, 1100.0, 0.0) 0 0
6 5.5M (1200.0, 1200.0, 0.0) 0 0
7 5M (1200.0, 1200.0, 0.0) 0 0
7 5.5M (1000.0, 1200.0, 0.0) 0 0
8 3M (1300.0, 1300.0, 0.0) 0 0
8 3.5M (1000.0, 1200.0, 0.0) 0 0
8 5M (1000.0, 1200.0, 0.0) 0 0
8 5.5M (800.0, 1200.0, 0.0) 0 0
8 7M (800.0, 1200.0, 0.0) 0 0
8 7.5M (600.0, 1200.0, 0.0) 0 0
8 9M (600.0, 1200.0, 0.0) 0 0
8 9.5M (400.0, 1200.0, 0.0) 0 0
8 11M (400.0, 1200.0, 0.0) 0 0
8 11.5M (200.0, 1200.0, 0.0) 0 0
7 9M (1000.0, 1200.0, 0.0) 0 0

7 9.5M (800.0, 1200.0, 0.0) 0 0
7 11M (800.0, 1200.0, 0.0) 0 0
7 11.5M (600.0, 1200.0, 0.0) 0 0
7 13M (600.0, 1200.0, 0.0) 0 0
7 13.5M (400.0, 1200.0, 0.0) 0 0
6 11M (1200.0, 1200.0, 0.0) 0 0
6 11.5M (1000.0, 1200.0, 0.0) 0 0
6 13M (1000.0, 1200.0, 0.0) 0 0
6 13.5M (900.0, 1200.0, 0.0) 0 0
6 15M (900.0, 1200.0, 0.0) 0 0
6 15.5M (600.0, 1200.0, 0.0) 0 0
5 13M (1400.0, 1200.0, 0.0) 0 0
5 13.5M (1300.0, 1200.0, 0.0) 0 0
5 15M (1300.0, 1200.0, 0.0) 0 0
5 15.5M (1200.0, 1200.0, 0.0) 0 0
5 17M (1200.0, 1200.0, 0.0) 0 0
5 17.5M (800.0, 1200.0, 0.0) 0 0

3.5 Network Model Settings

NETWORK-PROTOCOL IP
IP-ENABLE-LOOPBACK YES
IP-LOOPBACK-ADDRESS 127.0.0.1
IP-FRAGMENTATION-UNIT 2048
IP-QUEUE-NUM-PRIORITIES 3
IP-QUEUE-PRIORITY-INPUT-QUEUE-SIZE 50000
DUMMY-PRIORITY-QUEUE-SIZE NO
IP-QUEUE-PRIORITY-QUEUE-SIZE 50000
DUMMY-PRIORITY-WISE-IP-QUEUE-TYPE NO
IP-QUEUE-TYPE FIFO
ECN NO
IP-QUEUE-SCHEDULER STRICT-PRIORITY

3.6 Machine Configuration Scripts

3.6.1 Emulator/Simulator SharedNet Classic Configuration Script

```
route add default gw 192.168.3.1
echo "#Added by /root/setupEmulatorClassic.sh" > /etc/resolv.conf
echo "nameserver 4.2.2.1" >> /etc/resolv.conf
#MIP
route add 214.15.3.105 dev eth5
route add 214.15.3.106 dev eth5
route del 225.0.10.1
route add 225.0.10.1 gw 214.15.3.105
#MOBILE1
```

```
route add 214.15.3.103 dev eth6
route add 214.15.3.104 dev eth6
route del 225.0.10.2
route add 225.0.10.2 gw 214.15.3.103
#MOBILE2
route add 214.15.3.101 dev eth7
route add 214.15.3.102 dev eth7
route del 225.0.10.3
route add 225.0.10.3 gw 214.15.3.101
#MOBILE3
route add 214.15.3.99 dev eth8
route add 214.15.3.100 dev eth8
route del 225.0.10.4
route add 225.0.10.4 gw 214.15.3.99

#POP2
route add 10.1.101.1 dev eth2
route add 225.0.11.1 gw 10.1.101.1
#Mobile4
route add 10.1.101.2 dev eth3
route add 225.0.11.2 gw 10.1.101.2
#Mobile5
route add 10.1.101.3 dev eth4
route add 225.0.11.3 gw 10.1.101.3
#Mobile6
route add 10.1.101.4 dev eth11
route add 225.0.11.4 gw 10.1.101.4
```

3.6.2 Emulator/Simulator Machine SharedNet DTN Configuration Script

```
route add default gw 192.168.3.1
echo "#Added by /root/setupEmulatorDTN.sh" > /etc/resolv.conf
echo "nameserver 4.2.2.1" >> /etc/resolv.conf
#MIP
route add 214.15.3.105 dev eth5
route add 214.15.3.106 dev eth5
route del 225.0.10.1
route add 225.0.10.1 gw 214.15.3.106
#MOBILE1
route add 214.15.3.103 dev eth6
route add 214.15.3.104 dev eth6
route del 225.0.10.2
route add 225.0.10.2 gw 214.15.3.104
#MOBILE2
route add 214.15.3.101 dev eth7
route add 214.15.3.102 dev eth7
```

```
route del 225.0.10.3
route add 225.0.10.3 gw 214.15.3.102
#MOBILE3
route add 214.15.3.99 dev eth8
route add 214.15.3.100 dev eth8
route del 225.0.10.4
route add 225.0.10.4 gw 214.15.3.100
#POP2
route add 10.1.101.1 dev eth2
route add 225.0.11.1 gw 10.1.101.1
#Mobile4
route add 10.1.101.2 dev eth3
route add 225.0.11.2 gw 10.1.101.2
#Mobile5
route add 10.1.101.3 dev eth4
route add 225.0.11.3 gw 10.1.101.3
#Mobile6
route add 10.1.101.4 dev eth11
route add 225.0.11.4 gw 10.1.101.4
```

3.6.3 Client Machine (MOBILE2) DTN Configuration Script

```
#route add 225.0.10.2 gw 214.15.3.109
route add 214.15.3.106 gw 214.15.3.109
route add 214.15.3.104 gw 214.15.3.109
route add 214.15.3.100 gw 214.15.3.109
route add 234.1.1.1 gw 214.15.3.109
iptables -t nat -A OUTPUT -d 234.1.1.1 -j DNAT --to-destination 225.0.10.3
iptables -t nat -A PREROUTING -d 225.0.10.3 -j DNAT --to-destination 234.1.1.1
```

3.6.4 Client Machine (MOBILE2) Classic Configuration Script

```
route add 214.15.3.105 mask 255.255.255.255 214.15.3.109
route add 214.15.3.103 mask 255.255.255.255 214.15.3.109
route add 214.15.3.99 mask 255.255.255.255 214.15.3.109
route add 225.0.10.3 mask 255.255.255.255 214.15.3.109
```

3.7 Emulator/Simulator Running Kernel Modules (lsmod)

```
Module                  Size  Used by
nls_iso8859_1           4256  1
nls_cp437               5920  1
vfat                   11872  1
fat                    46652  1 vfat
usb_storage            72000  1
nvidia               8881444  28
agpgart                29896  1 nvidia
ppdev                   8676  0
```

```
parport_pc           32132  0
lp                   11012  0
parport              33256  3 ppdev,parport_pc,lp
button               6672  0
ac                   5188  0
battery              9636  0
ipv6                 226304  22
dm_snapshot          15552  0
dm_mirror            19152  0
dm_mod               50200  2 dm_snapshot,dm_mirror
sbp2                 20840  0
ieee1394             86904  1 sbp2
loop                 15048  0
snd_hda_intel        17332  0
snd_hda_codec        137856  1 snd_hda_intel
snd_pcm_oss          38368  0
snd_mixer_oss        15200  1 snd_pcm_oss
snd_pcm              68676  3 snd_hda_intel,snd_hda_codec,snd_pcm_oss
snd_timer            20996  1 snd_pcm
asix                 10400  0
usbnet               15464  1 asix
i2c_i801             7468  0
serio_raw            6660  0
snd                  47012  6
snd_hda_intel,snd_hda_codec,snd_pcm_oss,snd_mixer_oss,snd_pcm,snd_timer
shpchp               33024  0
pci_hotplug          28704  1 shpchp
nxge                 459456  0
pcspkr               3072  0
i2c_core             19680  2 nvidia,i2c_i801
psmouse              35016  0
soundcore            9248  1 snd
snd_page_alloc       10184  2 snd_hda_intel,snd_pcm
tsdev                7520  0
evdev                9088  1
rtc                  12372  0
sg                   31292  0
sr_mod               15876  0
cdrom                32544  1 sr_mod
ext3                 119336  1
jbd                  52456  1 ext3
mbcache              8356  1 ext3
sd_mod               19040  4
usbhid               37248  0
generic              4868  0 [permanent]
```

```
ahci                   17924  2
libata                 89396  1 ahci
e100                   32232  0
scsi_mod              124168  7 usb_storage,sbp2,sg,sr_mod,sd_mod,ahci,libata
piix                    9444  0 [permanent]
ide_core              110504  3 usb_storage,generic,piix
mii                     5344  2 asix,e100
ehci_hcd               28136  0
uhci_hcd               21164  0
usbcore               112644  7 usb_storage,asix,usbnet,usbhid,ehci_hcd,uhci_hcd
tg3                    94948  0
thermal                13608  0
processor              28840  1 thermal
fan                     4804  0
```

Appendix D - SharedNet Configuration Data

– The following information documents the software versions and Windows environment variables used in the MCTSSA testing:

Java version: 1.6.0_15

SharedNet version: 6.7.1.55.DTN

Environment variables:

```
# Needed unless localhost: address of BPA
DTNAPI_ADDR=IP address of BPA

# Or equivalent for Linux: main SN directory
SHAREDNET_HOME='C:\Progra~1\SharedNet'

# Force custody transfer on
SN_DTN_CUSTODY=true

# Or equivalent for Linux: Java to use for SN
SN_JAVA_HOME='C:\Progra~1\Java\jdk1.6.0_15'

# Turn on high level of logging
SN_LOG_LEVEL=DEBUG

# Or equivalent: location of MySQL; needed only for hosts running servers
SN_MYSQL_HOME='C:\Progra~1\mysql'

# IP address corresponding to host DNS entry
SN_PRIMARY_INTERFACE=IP address
```

Appendix E – MCTSSA Satcom Simulation Scripts

The scripts below were used to start the WANem and Linktrophy simulators that were used to simulate the two satellite hops for the MCTSSA testing. In the testing, all four satellite emulations were started simultaneously via Linux shell script that from the Administrative computer, so their start times were consistent for every test.

There was no way to seed the random number generators that governed the loss models in each emulator, but the overall aggregate loss statistics were as depicted in the plots shown in previous sections of this report.

The delay through a geostationary satellite due to one-way light time is about 300 ms, and the WANem simulators used this value. The Linktrophy 4500 simulators were programmed to additionally add modem processing and local network delays to the satellite light time delay, and per MCTSSA recommendation, a 600 ms delay was used.

Adding modem and other delays to the WANem emulation was unintentionally overlooked. It is not expected to change the conclusions or results, as the C2PC or SN systems are relatively insensitive to +/- 300ms differences in delay, but highly sensitive to disruption and loss, which were adequately modeled by these simulators.

3.1 WANem Script (wanemscript1.sh)

```
#!/bin/bash

STARTDELAY=20
# This variable is used by the stagin.sh script, and MUST be exported

echo "Starting 20 minute test run"

tc qdisc del dev eth1 root
tc qdisc add dev eth1 root handle 1: netem delay 300ms loss 0.5%
tc qdisc add dev eth1 parent 1:1 handle 10: htb default 1 r2q 10
tc class add dev eth1 parent 10: classid 0:1 htb rate 2048kbit ceil 2048kbit

tc qdisc del dev eth2 root
tc qdisc add dev eth2 root handle 1: netem delay 300ms loss 0.5%
tc qdisc add dev eth2 parent 1:1 handle 10: htb default 1 r2q 10
tc class add dev eth2 parent 10: classid 0:1 htb rate 2048kbit ceil 2048kbit
```

```
echo "Links configured, waiting for $STARTDELAY seconds to begin test"
sleep $STARTDELAY

echo "starting 20 minute run; no loss for 6 minutes"
tc qdisc change dev eth1 root handle 1: netem delay 300ms loss 0.5%
tc qdisc change dev eth2 root handle 1: netem delay 300ms loss 0.5%
sleep 360

echo "40% loss"
tc qdisc change dev eth1 root handle 1: netem delay 300ms loss 40%
tc qdisc change dev eth2 root handle 1: netem delay 300ms loss 40%
sleep 60

echo "80% loss"
tc qdisc change dev eth1 root handle 1: netem delay 300ms loss 80%
tc qdisc change dev eth2 root handle 1: netem delay 300ms loss 80%
sleep 60

echo "20% loss"
tc qdisc change dev eth1 root handle 1: netem delay 300ms loss 20%
tc qdisc change dev eth2 root handle 1: netem delay 300ms loss 20%
sleep 60

echo "0% loss"
tc qdisc change dev eth1 root handle 1: netem delay 300ms loss 0.5%
tc qdisc change dev eth2 root handle 1: netem delay 300ms loss 0.5%
sleep 60

echo "50% loss"
tc qdisc change dev eth1 root handle 1: netem delay 300ms loss 50%
tc qdisc change dev eth2 root handle 1: netem delay 300ms loss 50%
sleep 60

echo "20% loss"
tc qdisc change dev eth1 root handle 1: netem delay 300ms loss 20%
tc qdisc change dev eth2 root handle 1: netem delay 300ms loss 20%
sleep 60

echo "95% loss"
tc qdisc change dev eth1 root handle 1: netem delay 300ms loss 95%
tc qdisc change dev eth2 root handle 1: netem delay 300ms loss 95%
sleep 60

echo "100% loss"
tc qdisc change dev eth1 root handle 1: netem delay 300ms loss 100%
```

```
tc qdisc change dev eth2 root handle 1: netem delay 300ms loss 100%
sleep 60

echo "50% loss"
tc qdisc change dev eth1 root handle 1: netem delay 300ms loss 50%
tc qdisc change dev eth2 root handle 1: netem delay 300ms loss 50%
sleep 60

echo "20% loss"
tc qdisc change dev eth1 root handle 1: netem delay 300ms loss 20%
tc qdisc change dev eth2 root handle 1: netem delay 300ms loss 20%
sleep 60

echo "30% loss"
tc qdisc change dev eth1 root handle 1: netem delay 300ms loss 30%
tc qdisc change dev eth2 root handle 1: netem delay 300ms loss 30%
sleep 60

echo "10% loss"
tc qdisc change dev eth1 root handle 1: netem delay 300ms loss 10%
tc qdisc change dev eth2 root handle 1: netem delay 300ms loss 10%
sleep 60

echo "5% loss"
tc qdisc change dev eth1 root handle 1: netem delay 300ms loss 5%
tc qdisc change dev eth2 root handle 1: netem delay 300ms loss 5%
sleep 60

echo "0% loss"
tc qdisc change dev eth1 root handle 1: netem delay 300ms loss 0.5%
tc qdisc change dev eth2 root handle 1: netem delay 300ms loss 0.5%
sleep 60




echo "Test complete"
```

3.2 Linktrophy Start Script (C2PClink.exp)

```
#!/usr/bin/expect -f
#
# This Expect script was generated by autoexpect on Mon Feb 22
08:04:30 2010
# Expect and autoexpect were both written by Don Libes, NIST.
```

```
#
# Note that autoexpect does not guarantee a working script.  It
# necessarily has to guess about certain things.  Two reasons a script
# might fail are:
#
# 1) timing - A surprising number of programs (rn, ksh, zsh, telnet,
# etc.) and devices discard or ignore keystrokes that arrive "too
# quickly" after prompts.  If you find your new script hanging up at
# one spot, try adding a short sleep just before the previous send.
# Setting "force_conservative" to 1 (see below) makes Expect do this
# automatically - pausing briefly before sending each character.  This
# pacifies every program I know of.  The -c flag makes the script do
# this in the first place.  The -C flag allows you to define a
# character to toggle this mode off and on.

set force_conservative 0  ;# set to 1 to force conservative mode even if
                          ;# script wasn't run conservatively originally
if {$force_conservative} {
        set send_slow {1 .1}
        proc send {ignore arg} {
                sleep .1
                exp_send -s -- $arg
        }
}

#
# 2) differing output - Some programs produce different output each time
# they run.  The "date" command is an obvious example.  Another is
# ftp, if it produces throughput statistics at the end of a file
# transfer.  If this causes a problem, delete these patterns or replace
# them with wildcards.  An alternative is to use the -p flag (for
# "prompt") which makes Expect only look for the last line of output
# (i.e., the prompt).  The -P flag allows you to define a character to
# toggle this mode off and on.
#
# Read the man page for more info.
#
# -Don


set timeout -1
spawn telnet 192.168.3.98
match_max 100000
expect -exact "Trying 192.168.3.98...\r\r
Connected to 192.168.3.98.\r\r
```

```
Escape character is '^\]'.\r\r
\r\r
linktropy login: "
send -- "admin\r"
expect -exact "admin\r
C2PC> "
send -- "link 1 set bandwidth 2000000 delay constant 600 loss 0%"
expect -exact "link 1 set bandwidth 2000000 delay constant 600 loss 0%"
send -- "\r"
expect -exact "\r\r
C2PC> "
sleep 360
send -- "link 1 set bandwidth 2000000 delay constant 600 loss 0%"
expect -exact "link 1 set bandwidth 2000000 delay constant 600 loss 0%"
send -- "   "
expect -exact "◘←\[K"
send -- "   "
expect -exact "◘←\[K"
send -- "30%\r"
expect -exact "30%\r\r
C2PC> "
sleep 60
send -- "link 1 set bandwidth 2000000 delay constant 600 loss"
expect -exact "link 1 set bandwidth 2000000 delay constant 600 loss"
send -- " 0.5%\r"
expect -exact " 0.5%\r\r
C2PC> "
sleep 60
send -- "link 1 set bandwidth 2000000 delay constant 600 loss"
expect -exact "link 1 set bandwidth 2000000 delay constant 600 loss"
send -- " 10%\r"
expect -exact " 10%\r\r
C2PC> "
sleep 60
send -- "link 1 set bandwidth 2000000 delay constant 600 loss"
expect -exact "link 1 set bandwidth 2000000 delay constant 600 loss"
send -- " 20%\r"
expect -exact " 20%\r\r
C2PC> "
sleep 60
send -- "link 1 set bandwidth 2000000 delay constant 600 loss"
expect -exact "link 1 set bandwidth 2000000 delay constant 600 loss"
send -- " 100%\r"
expect -exact " 100%\r\r
C2PC> "
```

```
sleep 120
send -- "link 1 set bandwidth 2000000 delay constant 600 loss"
expect -exact "link 1 set bandwidth 2000000 delay constant 600 loss"
send -- " 20%\r"
expect -exact " 20%\r\r
C2PC> "
sleep 60
send -- "link 1 set bandwidth 2000000 delay constant 600 loss"
expect -exact "link 1 set bandwidth 2000000 delay constant 600 loss"
send -- " 10%\r"
expect -exact " 10%\r\r
C2PC> "
sleep 60
send -- "link 1 set bandwidth 2000000 delay constant 600 loss"
expect -exact "link 1 set bandwidth 2000000 delay constant 600 loss"
send -- " 0.5%\r"
expect -exact " 0.5%\r\r
C2PC> "
sleep 60
send -- "link 1 set bandwidth 2000000 delay constant 600 loss"
expect -exact "link 1 set bandwidth 2000000 delay constant 600 loss"
send -- " 80%\r"
expect -exact " 80%\r\r
C2PC> "
sleep 120
send -- "link 1 set bandwidth 2000000 delay constant 600 loss"
expect -exact "link 1 set bandwidth 2000000 delay constant 600 loss"
send -- " 30%\r"
expect -exact " 30%\r\r
C2PC> "
sleep 60
send -- "link 1 set bandwidth 2000000 delay constant 600 loss"
expect -exact "link 1 set bandwidth 2000000 delay constant 600 loss"
send -- " 15%\r"
expect -exact " 15%\r\r
C2PC> "
sleep 60
send -- "link 1 set bandwidth 2000000 delay constant 600 loss"
expect -exact "link 1 set bandwidth 2000000 delay constant 600 loss"
send -- " 5%\r"
expect -exact " 5%\r\r
C2PC> "
sleep 60
send -- "link 1 set bandwidth 2000000 delay constant 600 loss"
expect -exact "link 1 set bandwidth 2000000 delay constant 600 loss"
```

```
send -- " 0%\r"
expect -exact " 0%\r\r
C2PC> "
send -- "exit\r"
expect eof
```